SOLUCIÓN DEL PROBLEMA DE LA MOCHILA BIDIMENSIONAL IRRESTRICTA USANDO OPTIMIZACIÓN CON CÚMULO DE PARTÍCULAS

FABIÁN FERNANDO MONTOYA GARCÍA

PROPUESTA DE PROYECTO DE GRADO
PARA OPTAR AL TITULO DE MAGÍSTER EN INGENIERÍA ELÉCTRICA

DIRECTOR
PH.D. RAMÓN ALFONSO GALLEGO RENDÓN

MAESTRÍA EN INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERIAS: ELÉCTRICA, ELECTRÓNICA FÍSICA Y CIENCIAS COMPUTACIONALES
UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PEREIRA, JUNIO DE 2017

Resumen

El problema de la mochila bidimensional irrestricta es de interés tanto académico como industrial. Este es considerado clásico dentro de la investigación operativa debido a su alta complejidad matemática y computacional.

En este trabajo se presenta el estado del arte del problema de la mochila bidimensional guillotinada irrestricta, considerando la posibilidad de que las piezas tengan o no valores asociados y que las piezas puedan o no rotar 90°. Se propone un modelo matemático basado en el aplicado por los grupos de investigación en estos problemas. Se desarrolla un tipo de codificación para ser aplicada en este problema y resolverla mediante técnicas aproximadas como lo son las heurísticas y metaheurísticas. Se implementa una metodología basada en técnicas metaheurísticas bien conocidas como: optimización con cúmulo de partículas, recocido simulado y algoritmos genéticos, las cuales van de la mano con heurísticas del problema, aplicando nuevos métodos de solución eficientes en cuanto a tiempo y calidad de las respuestas.

Para comprobar la eficiencia de las metodologías presentadas se tomaron casos de prueba de la literatura especializada, donde se analizan y comparan los métodos de solución presentados con los del estado del arte de los problemas, obteniéndose resultados de excelente calidad y nunca antes reportados en la literatura.

Abstract

The two-dimensional unconstrained knapsack problem is of big interest for the academy and the industry. It is considered a classic problem in the operations research, due to its highly mathematical and computational complexity.

In this thesis we show the state-of-the-art of two-dimensional unconstrained single guillotine knapsack problem, weighted and unweigthed versions and with and without items rotations of 90°. We propose for these problems a mathematical model based on the recognized by the academic community. We develop an appropriate encoding of the problem to work on it by approximated techniques, such as heuristics and metaheuristics. We implement a methodology based on metaheuristics techniques well known as: particle swarm optimization, simulated annealing and genetic algorithms, these go along with heuristics of the problem, generating new and efficient methods of solution in relation to time and quality of the responses.

To check the efficiency of the presented methodologies, case of studies were taken from specialized literature, where it could be analyzed and compared the presented solution methods with the state-of-the-art of the problems, we obtained results of excellent quality and never reported in the literature.

ÍNDICE GENERAL

Pág.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Tabla **Título de la tabla** **Pág.**

Capítulo 1.

INTRODUCCIÓN

1.1. Definición del problema

El problema de la mochila irrestricta bidimensional guillotinada (U2DGSKP) del inglés *unconstrained two-dimensional guillotineable single knapsack problem*, es un problema de corte que se presenta cuando el material a ser utilizado es una pieza rectangular (hoja de material) donde se deben ubicar piezas rectangulares más pequeñas de las que se conoce el tamaño y un costo asociado, el objetivo es maximizar el valor de las piezas cortadas, sin sobreponer las piezas y sin sobrepasar los límites de la hoja de material.

Las características de este problema son las siguientes:

i) El costo asociado puede o no estar relacionado con el área de la pieza a ser ubicada; si el costo es igual al área de la pieza se está resolviendo el problema sin pesos (*unweighted version*) y si el costo es diferente del área del ítem el problema a resolver será el problema con pesos (*weighted version*).

ii) La orientación de las piezas a ser ubicadas, es decir, una pieza de alto h y ancho w es diferente de una pieza de longitud w y alto h (*without rotation*). Si se considera que las dimensiones (h, w) y (w, h) representan las dimensiones de la misma pieza, se está abordando un problema con rotación (*with rotation*).

iii) Los patrones de corte son del tipo guillotina, cada corte produce dos sub-rectángulos. Los cortes van de un extremo al otro del rectángulo original.

iv) No existe un límite máximo del número de piezas a cortar de cada tipo. (*unconstrained version*)

Se ha demostrado que este problema es NP-Duro debido a su alta complejidad matemática [1]. Existen cuatro variantes de los mismos dependiendo del costo asociado a las piezas y de la orientación de las piezas: con pesos y con rotación, con pesos y sin rotación, sin pesos y con rotación y sin pesos y sin rotación. Este conjunto de problemas es aplicable en diversos sectores de la economía, entre los que destacan los sectores de industria, transporte y comercio. Así por ejemplo, su aplicación se presenta en la industria textil, metalmecánica, papelera, vidriera, cuerera, transporte y almacenaje de mercancías, entre otras.

1.2. Motivación

Los problemas de corte y empaquetamiento son de interés tanto industrial como académico, avanzar en la temática de corte y empaquetamiento óptimo bidimensional guillotinado puede maximizar el uso de las materias primas, factor que incide directamente en el costo final del producto y son de aplicación directa en empresas de corte, transporte y almacenamiento de materiales, esto representa una estrategia de mejoramiento que redunda en el nivel de ganancias de una organización de esta naturaleza.

Este estudio está basado en una revisión del estado del arte de los problemas clásicos de corte y empaquetamiento presenta una nueva metodología para la solución del problema de la mochila bidimensional irrestricta guillotinada, con base en esta pueden ser abordadas muchas otras variantes para este tipo de problemas.

La industria se ha concientizado de la importancia de la investigación y desarrollo en las áreas de investigación operativa e ingeniería de la producción, la carencia en el mercado de aplicativos informáticos que permitan al sector industrial disponer de herramientas que den solución eficiente a sus problemáticas de interés ha sido uno de los desafíos de este proyecto, en este orden de ideas se ha tomado como herramienta para implementar la solución del problema planteado las técnicas metaheurísticas de optimización ya que han mostrado un excelente desempeño en la solución de problemas complejos en diversas áreas de la ingeniería.

1.3. Objetivos

GENERAL

Desarrollar una metodología de solución para el problema de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas, usando un algoritmo de optimización que combina las principales características de las técnicas metaheurísticas de optimización cúmulo de partículas, recocido simulado y algoritmos genéticos.

ESPECÍFICOS

- Revisar el estado del arte de los modelos matemáticos que representan el problema de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas.

- Revisar el estado del arte de las técnicas de solución utilizadas en la solución de este tipo de problemas.

- Proponer una codificación eficiente que permita la implementación de técnicas heurísticas y metaheurísticas de optimización, en la solución del problema de empaquetamiento.

- Desarrollar una nueva metodología para la solución del problema de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas.

- Validar las metodologías propuestas con casos de prueba de la literatura especializada.

1.4. Alcance y contribuciones

Las principales contribuciones de esta tesis son las siguientes:

- Se desarrollará una metodología para la solución el problema de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas.

- Se pretende que los resultados obtenidos por la metodología presentada en esta tesis superen a muchos de los que se encuentran propuestos en la literatura especializada.

- Todas las implementaciones realizadas en este trabajo se compararan contra técnicas representativas del estado del arte de optimización con técnicas exactas, heurísticas y metaheurísticas, con el fin de concluir si las metodologías desarrolladas en esta tesis presentan resultados de igual o mejor o peor calidad respecto a los conocidos.

1.4.Estructura general del documento

La organización de este documento es la siguiente:

En el capítulo 2 se plantea el problema de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas.

En el capítulo 3 se presentan y se explican los modelos matemáticos de los problema de empaquetamiento óptimo bidimensional de piezas rectangulares acogidos por la comunidad académica.

En el capítulo 4 se realiza una descripción de las técnicas de optimización: recocido simulado y optimización con cúmulo de partículas.

En el capítulo 5 se presenta la codificación del problema y la adaptación de las técnicas de optimización combinatoria.

En el capítulo 6 se realiza un análisis formal de los resultados obtenidos por las diferentes metodologías presentadas.

En el capítulo 7 se presentan las conclusiones y recomendaciones.

Capítulo 2.

PLANTEAMIENTO DEL PROBLEMA DE LA MOCHILA BIDIMENSIONAL IRRESTRICTA DE PIEZAS RECTANGULARES

2.1. Generalidades de los problemas de la mochila

(Washer *et al*., 2007) caracterizan los problemas de la mochila como problemas de empaquetamiento con un surtido de piezas fuertemente heterogéneo que debe ser ubicado en un conjunto de contenedores. La disponibilidad de contenedores es limitada de tal manera que no es suficiente para ubicar todas las piezas. El valor de las piezas embaladas debe ser maximizado.

En (Washer *et al*., 2007) es generada la categoría *single knapsack problem* (SKP), dentro de esta se encuentran problemas como la mochila clásica (*One-Dimensional Knapsack Problem*), también llamada mochila 0-1 (Martello *et al*., 2000; Martello y Toth, 1990) la cual requiere ubicar un conjunto de piezas de volúmenes dados y costos asociados dentro de una (*single*) mochila con un límite de volumen y donde el objetivo es maximizar los costos asociados de las piezas embaladas.

(Caprara y Monaci, 2004; Healy y Moll, 1996) presentan el problema de una mochila bidimensional ortogonal, donde un conjunto de pequeños rectángulos de tamaño y beneficio asociado dado tiene que ser cortado de un gran rectángulo, con el fin de maximizar el beneficio de las piezas cortadas (versión con pesos), usualmente el beneficio o valor asociado de las pequeñas piezas puede asumirse proporcional al tamaño de estas, siendo el objetivo equivalente a minimizar el espacio inutilizado del gran rectángulo (versión sin pesos) (Fekete y Schepers, 1997). Estos últimos son los problemas de estudio propuesto en este trabajo, con algunas restricciones y variantes adicionales:

- Solo se permite el uso de cortes tipo guillotina, es decir, los cortes van de un extremo al otro del rectángulo original (guillotinado).
- No existe un límite máximo del número de piezas a cortar de cada tipo (irrestricto).
- La orientación de las piezas a ser ubicadas, es decir, una pieza de alto h y ancho w es diferente de una pieza de longitud w y alto h (sin rotación). Si se considera que las dimensiones (h, w) y (w, h) representan las dimensiones de la misma pieza, se está abordando un problema con rotación (con rotación).

Por lo tanto, en este estudio se pretende resolver los problemas de la mochila bidimensional irrestricta guillotinada, con y sin pesos asociados a las piezas, con y sin rotación de las piezas. En la tipología propuesta por (Washer *et al*., 2007) los problemas trabajados en este estudio recaen en el tipo SLOPP (de la sigla en inglés *Single Large Object Placement Problem*).

Por último, dentro de esta categoría también se encuentran: el problema de la mochila tridimensional presentada por (Pisinger, 2002) bien conocido como problema de carga de contenedores, en el cual cajas rectangulares deben ser empacadas dentro de un contenedor tridimensional rectangular, donde el objetivo es maximizar el valor de las

cajas embaladas y el problema de la mochila n-dimensional, donde las piezas son representadas en n-dimensiones geométricas (Bischoff y Marriott, 1990; Scheithauer, 1999; Bortfeldt y Gehring, 2001).

2.2. Aplicaciones del problema de la mochila en la industria

Este problema es aplicable en diversos sectores de la economía, entre los que destacan los sectores de industria, transporte y comercio. Así por ejemplo, sus aplicaciones se pueden observar en industrias de perfiles metálicos, corte de maderas, papel, plástico, piel, aluminio o vidrio en donde los componentes rectangulares tienen que ser cortados desde grandes hojas de material; en un contexto de depósitos o almacenes las mercancías deben ser ubicadas en estanterías; en periódicos artículos y avisos tienen que ser organizados en páginas. La característica especial de estas aplicaciones, son las unidades de representación del material (rectángulos), y tienen funciones objetivo comunes que consisten en que se maximice el beneficio asociado de las piezas cortadas (con pesos) o que se minimice el área inutilizada (sin pesos). Diferente a otros contextos como las empresas papeleras o de industrias textiles donde la unidad de material es un rollo y la función objetivo es cortar todos los ítems usando la mínima dimensión del rollo (*two-dimensional strip packing problem*).

En la industria textil, el proceso de corte se realiza después del diseño de las prendas, el inventario de las telas necesarias y la descripción de los patrones que se desean obtener de cada una de ellas.

En el corte intervienen una o más personas especializadas que son quienes van distribuyendo y cortando los patrones sobre la tela. Para automatizar y optimizar este proceso, el sistema a desarrollar debería cumplir:

- El desperdicio de telas obtenida por el sistema debería ser igual o menor que la obtenida por un experto (humano especializado). El desperdicio producido por un profesional del corte es de aproximadamente el 30% de la tela original.
- El tiempo de ejecución para la obtención de un resultado admisible, no debe en ningún caso provocar una demora en el proceso global de producción, pues en tal caso, no proporcionaría ningún beneficio notable para la industria en cuestión.

A la hora de resolver el problema, se deben considerar y aplicar todas las restricciones referentes a la legalidad de los patrones. Por ejemplo, algunos patrones tienen orientación y ésta debe seguir la dirección del material. Además, para los problemas con pesos asociados a las piezas, donde los materiales cuya calidad no es la misma en toda su superficie, por ejemplo, con el cuero y la madera, se suele aprovechar las zonas de mejor calidad para fabricar las piezas de mayor valor.

2.3. Introducción histórica de los problemas de corte y empaquetamiento

Los problemas de empaquetamiento y corte óptimo (*cutting/packing problems*) son considerados clásicos dentro de la investigación de operaciones. Este se comenzó a plantear y estudiar hace más de 6 décadas. Primero fue estudiado por (Kantorovich, 1939) y luego (Gilmore y Gomory, 1961) trabajaron problemas reales a través de la programación lineal.
Gran cantidad de artículos con diferentes aproximaciones y estudios diversos acerca de las posibles variaciones del problema siguen apareciendo.

(Sweeney y Paternoster, 1992) realizaron una completísima recopilación de documentación entre 1940 y 1990 corte y empaquetamiento, (Washer *et al.*, 2007) entre 1990 y 2005 y por último (Lodi *et al.*, 2002) presentan los avances recientes en esta temática.

2.4. Patrones de corte bidimensional

(Morabito y Morales, 1998) clasifican de los patrones de empaquetamiento: patrones guillotina y no guillotina. Un patrón es de tipo guillotina si se puede obtener por sucesivos cortes de tipo guillotina (ver figura 2.1). Un patrón es no guillotina si es obtenido por sucesivos cortes de guillotina y no guillotina (ver figura 2.2).

Un corte es de tipo guillotina si cuando se aplica sobre un rectángulo produce dos nuevos rectángulos, es decir, si el corte va de un extremo a otro del rectángulo original; en otro caso se denomina del tipo no guillotina.

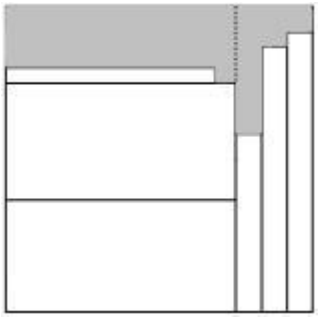

Figura 2.1. Corte guillotina.

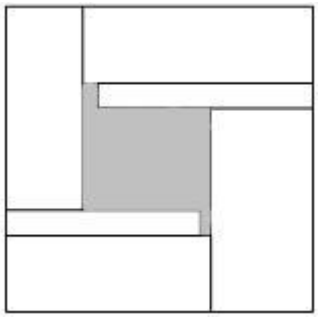

Figura 2.2. Corte no guillotina.

2.5. Extensión a polígonos

(Jakobs, 1996) propone para el problema de corte bidimensional de polígonos una reducción del problema a uno de empaquetamiento de rectángulos, al encontrar los rectángulos con la mínima área que contengan los polígonos, rotándolas alrededor del centro de gravedad (ver figura 2.3). Después de haber encajado cada polígono en rectángulos se resuelve el problema de empaquetamiento bidimensional rectangular.

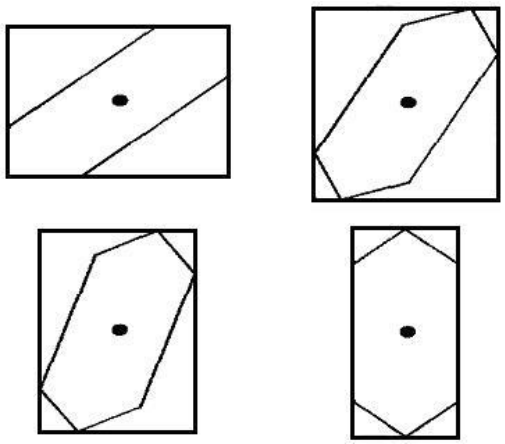

Figura 2.3. Empaquetamiento de un polígono en un rectángulo con mínima área.

(Jakobs, 1996) propone un algoritmo genético y lo compara respecto a la heurística constructiva esquina inferior izquierda (Chazelle, 1983, *bottom left*). Obteniendo excelentes resultados, las figuras 2.4 y 2.5 ilustran la diferencia entre los patrones de corte del algoritmo genético y la heurística constructiva.

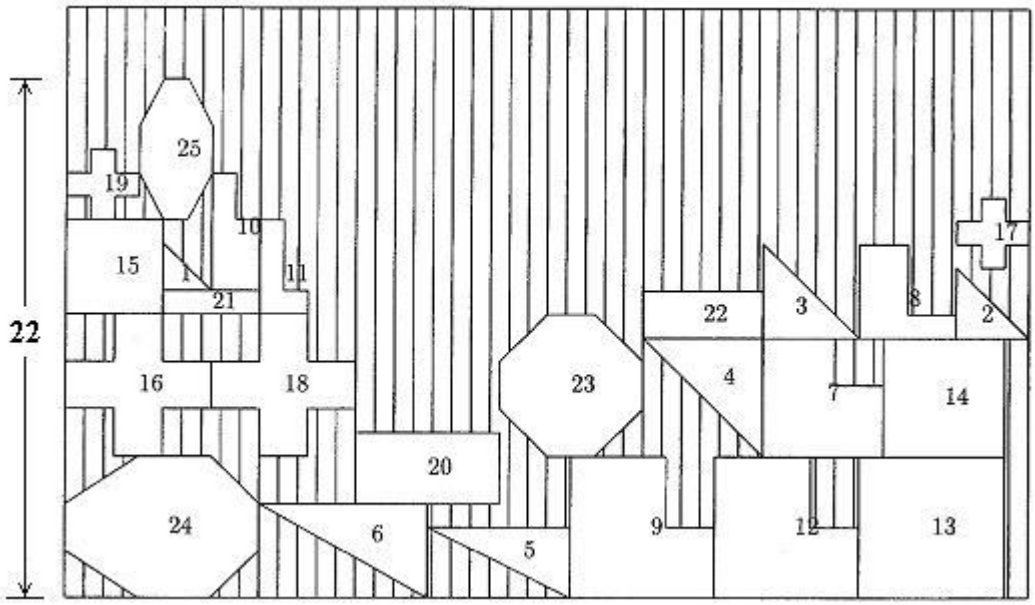

Figura 2.4. Respuesta obtenida con algoritmo *bottom left* (Altura =22).

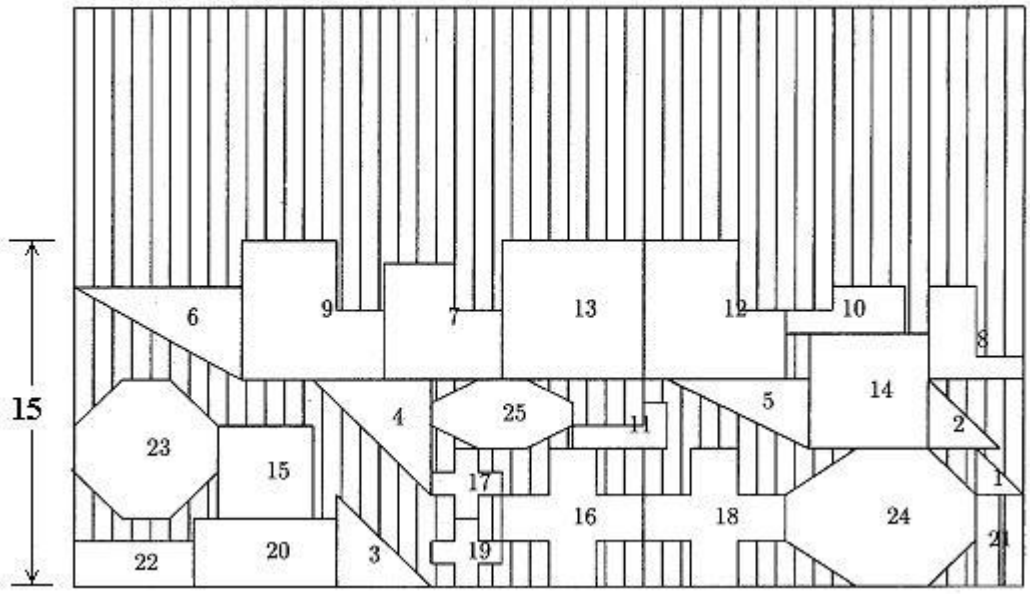

Figura 2.5. Respuesta obtenida con algoritmo genético (Altura=15).

2.6. Aproximaciones al problema de la mochila bidimensional

(Gilmore y Gomory, 1965; 1966) proponen un algoritmo recursivo exacto sobre la base de la programación dinámica para resolver el problema. Este algoritmo es aplicable a las versiones con y sin pesos. (Herz, 1972) propone un método de búsqueda recursiva de árbol, su método es más eficaz que el algoritmo de Gilmore y Gomory para el problema sin pesos, pero no se aplica a los casos con pesos. (Beasley, 1985) presenta un enfoque de basado en un algoritmo heurístico y programación dinámica, el cual es una versión

modificada del algoritmo de Gilmore y Gomory, este prueba su metodología con problemas de mediana escala (hasta 50 tipos de piezas), obteniendo buenos resultados.

(Morabito *et al*., 1992) desarrollaron la heurística DH (de las siglas en ingles *Depth-first search y Hill-climbing strategies*) basado en un proceso de búsqueda primero en profundidad y estrategias de aceptación de empeoramientos. Estas dos estrategias son comúnmente usadas en métodos de inteligencia artificial (Gallego *et al*., 2007, Gallego *et al*., 2008). El algoritmo KD (de las siglas en inglés *Knapsack problem* usando *Dynamic programming*) fue presentado por (Fayard y Zissimopoulos, 1995) para resolver el problema basado en la solución de problemas de la mochila unidimensional, resultando eficiente para problemas de gran tamaño. Este algoritmo también usa una estructura de grafo, pero solo considera el primer nivel del árbol, a diferencia del DH que consiste en desarrollar un árbol limitado por un parámetro de profundidad (Herz, 1972; Christofides y Whitlock, 1977; Morabito *et al*., 1992).

(Hifi, 2001) presenta un algoritmo híbrido que combina el algoritmo DH y KD, esto permite desarrollar un algoritmo general para el problema de la mochila bidimensional irrestricta guillotinada. Este prueba su algoritmo para problemas de gran escala, tanto en problemas con pesos como sin pesos, obteniendo excelente resultados.

(Hifi y Zissimopoulos, 1996) proponen un algoritmo recursivo exacto usando programación dinámica basada en eficientes límites inferiores y superiores para resolver el problema irrestricto. (G y Kang, 2002) proponen una mejora al algoritmo de (Hifi y Zissimopoulos, 1996) usando una cota superior más eficiente. Este es actualmente uno de los algoritmos exactos con mejor desempeño para resolver el problema irrestricto.

Existen dos técnicas generales utilizadas para resolver los problemas restrictos: *top-down* y *bottom-up* . (Christofides y Whitlock, 1977) propusieron originalmente el enfoque de *top-down*, el cual genera todos los posibles patrones de solución en forma recursiva, dividiendo cada placa en dos nuevas. La mayoría de los algoritmos exactos para el problema irrestricto utilizan este enfoque. El enfoque de *bottom-up* genera todos los posibles patrones de corte mediante la combinación de dos patrones horizontalmente o verticalmente construidos a partir de otros dos patrones. El enfoque *bottom-up* requiere una gran cantidad de memoria, razón por la cual la implementación de un algoritmo de este tipo es poco atractivo. Sin embargo, (G *et al*., 2003) propusieron un algoritmo que parte de una solución inicial de buena calidad y utiliza el algoritmo constructivo *bottom-up* como estrategia para generar las ramas, disminuyendo el número de nodos a explorar.

Los problemas de corte y empaquetamiento revisten una alta complejidad matemática, estos son considerados NP-Duros en un fuerte sentido, luego de ser demostrados a través del caso especial donde todos los ítems tienen la misma longitud y diferente altura, el bien conocido problema de empaquetamiento unidimensional (*one-dimensional bin packing*). El problema de empaquetamiento óptimo de una dimensión fue probado NP-Duro por (Garey y Johnson, 1979) y (Martello *et al*., 2003). El que un problema este catalogado en esta categoría no significa que no puede resolverse, sino que se deben proponer algoritmos de solución que exploten de forma eficiente la estructura matemática del mismo para que se encuentren soluciones a la mayoría de las instancias del problema en tiempos de ejecución relativamente pequeños.

A diferencia del problema restricto la mochila irrestricta no presenta aproximaciones metaheurísticas y la mayoría de los grupos de investigación han optado por el uso de técnicas exactas. En este estudio se propone el uso de tres metaheurísticas para el desarrollo de un algoritmo que reúna las características principales de las técnicas de cúmulo de partículas, algoritmos genéticos y recocido simulado (PSO, GA y SA, de las siglas en inglés *particle swarm optimization, genetic algorithms* y *simulated annealing* respectivamente).

2.7. Resumen

Los problemas propuestos hacen parte de problemas clásicos y de gran interés dentro del área de la investigación operativa, dado que son aplicables en diversos sectores de la economía, entre los que destacan los sectores de industria, transporte y comercio. Así por ejemplo, su aplicación se presenta en la industria textil, metalmecánica, papelera, vidriera, cuerera, transporte y almacenaje de mercancías, entre otras. La alta complejidad matemática y computacional de estos problemas ha llevado que sean solucionados desde diferentes áreas de la optimización como lo son la optimización exacta y la optimización combinatoria, una lista de trabajos y contribuciones sobre esta temática ha sido expresada anteriormente.

Se pretende trabajar el problema desde la optimización combinatoria haciendo uso de herramientas como las técnicas metaheurísticas de optimización. Este estudio propone el uso de tres técnicas metaheurísticas: optimización con cúmulos de partículas, algoritmos genéticos y recocido simulado para desarrollar un método de solución a través de la creación un de algoritmo híbrido de optimización, para dar solución a:

- El problema de la mochila bidimensional irrestricta guillotinada con pesos asociados a las piezas, con y sin rotación de piezas.
- El problema de la mochila bidimensional irrestricta guillotinada sin pesos asociados a las piezas, con y sin rotación de piezas.

Capítulo 3.

MODELOS MATEMÁTICOS DE LOS PROBLEMAS

3.1. Introducción

Distintos modelos matemáticos para representar el mismo problema han sido propuestos, con el fin de encontrar mejoras en la solución a través de un modelamiento eficiente, es por esto que en este estudio se dedica un capítulo a los modelos matemáticos que representan el problema de la mochila bidimensional irrestricta guillotinada con y sin rotación de piezas.

(Gilmore y Gomory, 1961; 1965) presentan el primer modelo matemático para el problema de corte y empaquetamiento unidimensional. (Gilmore y Gomory, 1965; 1966) estudian cuidadosamente el problema de la mochila cuando este es aplicado para el corte de piezas en una y dos dimensiones.

(Biro y Boros, 2005) caracterizan los patrones no guillotina usando teoría de grafos. (Beasley, 1986) estudió el problema de la mochila sin restricciones de corte tipo guillotina. (Beasley, 1986) propone un modelo de programación lineal entera para determinar las coordenadas discretas a las cuales los ítems pueden ser ubicados. Un modelo similar fue introducido por (Hadjiconstantinou y Christofides, 1995). (Scheithauer y Terno, 1993) proponen modelos para empaquetamiento de polígonos convexos y no convexos.

(Fekete y Schepers, 1997) usan la teoría de grafos para determinar un empaquetamiento factible sin sobreponer las piezas, pero no consideraron las restricciones de corte tipo guillotina. (*Lodi et al*., 2002; 2004) presentan un modelo que maneja patrones formados por estantes (niveles), este modelo considera explícitamente restricciones de corte tipo guillotina pero solo una parte de los patrones guillotina son considerados. Recientemente (Beasley, 2004) presenta una nueva formulación de corte no guillotina. Una buena revisión de los modelos existentes está disponible en (Lodi *et al*., 2004).

Un modelo matemático entero-mixto para el problema de empaquetamiento tridimensional en contenedores, con número polinomial de variables y restricciones, es presentado por (Chen *et al*., 1995), este modelo puede ser visto como una extensión a la técnica de modelamiento propuesta por (Onodera *et al*., 1991). Para un problema de ubicación de bloques bidimensionales, el modelo se basa en la enumeración de todas posibles ubicaciones relativas de cada par de piezas. Los experimentos computacionales presentados en (Chen *et al*., 1995) muestran sin embargo que el modelo propuesto es muy ineficiente en la solución de instancias prácticas de empaquetamiento. Además, no existe una manera fácil de adaptar las restricciones tipo guillotina. La misma técnica fue usada por (Daniels *et al*., 1994) para modelar el problema de empaquetamiento general (bidimensional) de polígonos, en este mismo caso, el uso directo del modelo prueba ser ineficiente en la práctica.

Como se muestra en la revisión bibliográfica anterior no existen modelos matemáticos que describan completamente los problemas de interés en este estudio.

(Ben *et al*., 2008) presentan un modelo basado en la caracterización y modelamiento de los patrones guillotina para el problema de empaquetamiento en rollos infinitos (*strip packing problem*), este es adaptado para representar el problema aquí propuesto.

3.2. Modelo matemático de la mochila

Se presenta un modelo de programación lineal entera mixta para el problema de empaquetamiento óptimo de la mochila bidimensional irrestricta basado en la caracterización de los patrones de corte tipo guillotina y el uso de coordenadas donde pueden ser ubicadas las piezas.

Se asume que existen n tipos de piezas que pueden ser cortadas de una mochila de ancho W y altura H. Para un patrón de corte donde la esquina inferior izquierda de cada pieza k ($k = 1, 2, \ldots, n$) es ubicada en las coordenadas (α_k, β_k), se pueden obtener dos series $(x_1, x_2, \ldots, x_n)$ y $(y_1, y_2, \ldots, y_n)$ por un ordenamiento decreciente de las series $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ y $(\beta_1, \beta_2, \ldots, \beta_n)$ respectivamente, se obtiene $0 \le x_1 \le x_2 \le \ldots \le x_n \le W$ y $0 \le y_1 \le y_2 \le \ldots \le y_n \le H$.

Para obtener un modelo lineal entero, se usaran el siguiente conjunto de variables binarias para representar la ubicación de las piezas en la mochila:

$z_{i,j,k} = \begin{cases} 1 & \text{si la pieza } k \text{ es empacada en } (i, j) \\ 0 & \text{de lo contrario} \end{cases}$

Las siguientes variables de decisión intermedias también son necesarias para garantizar que no existan traslapes entre piezas:

$u_{i,j,i'} = \begin{cases} 1 & \text{si la pieza en } (i, j), \text{ no excede (horizontalmente) } x_{i'} \text{ con } i' > i \\ 0 & \text{de lo contrario} \end{cases}$

$v_{i,j,j'} = \begin{cases} 1 & \text{si la pieza en } (i, j), \text{ no excede (verticalmente) } y_{j'} \text{ con } j' > j \\ 0 & \text{de lo contrario} \end{cases}$

Los siguientes tres conjuntos de variables binarias son necesarios para garantizar las restricciones guillotina:

$\mu_{i_1,i',j_1,j_2} = \begin{cases} 1, & \text{si no hay pieza entre } (i_1, j_1) \text{ y } (i'-1, j_2) \text{ que exceda } x_{i'} \text{ (consecuentemente, un corte vertical en } x_{i'} \text{ no cruza ninguna pieza empacada entre } (i_1, j_1) \text{ y } (i'-1, j_2)), \ i' > i_1 \\ 0, & \text{de lo contrario} \end{cases}$

$\nu_{i_1,i_2,j',j_1} = \begin{cases} 1, & \text{si no hay pieza entre } (i_1, j_1) \text{ y } (i_2, j'-1) \text{ que exceda } y_{j'} \text{ (consecuentemente, un corte horizontal en } y_{j'} \text{ no cruza ninguna pieza empacada entre } (i_1, j_1) \text{ y } (i_2, j'-1)), \ j' > j_1 \\ 0, & \text{de lo contrario} \end{cases}$

$u_{i_1,i_2,j_1,j_2} = \begin{cases} 1, & \text{si existe mínimo una pieza empacada entre } (i_1, j_1) \text{ y } (i_2, j_2) \\ 0, & \text{de lo contrario} \end{cases}$

La formulación completa del problema de empaquetamiento óptimo guillotinado de la mochila bidimensional irrestricta con pesos asociados a las piezas y sin rotación, es la siguiente:

Maximizar $\sum_{i=1}^{n}\sum_{j=1}^{n} c_k z_{i,j,k} \quad \forall k,$ (3.1)

(3.2)

sujeto a $0 \le x_1 \le x_2 \le \ldots \le x_n$,

$0 \le y_1 \le y_2 \le \ldots \le y_n$, (3.3)

$\sum_{i=1}^{n}\sum_{j=1}^{n} z_{i,j,k} \le 1 \quad \forall k,$ (3.4)

$\sum_{i=1}^{n}\sum_{k=1}^{n} z_{i,j,k} \le 1 \quad \forall j,$ (3.5)

$\sum_{j=1}^{n}\sum_{k=1}^{n} z_{i,j,k} \le 1 \quad \forall l,$ (3.6)

$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge \left(u_{i,j,i'} - 1\right) W \quad \forall i, \forall j, \forall i' > i,$ (3.7)

$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge u_{i,j,i'} W \quad \forall i, \forall j, \forall i' > i,$ (3.8)

$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge \left(v_{i,j,i'} - 1\right) H \quad \forall i, \forall j, \forall j' > j,$ (3.9)

$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge u_{i,j,j'} H \quad \forall i, \forall j, \forall j' > j,$ (3.10)

$\left(1 - a_{i_1,i_2,j_1,j_2}\right) n \ge \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} - 1 \quad \forall l_1, \forall j_1, \forall l_2 > l_1, \forall j_2 > j_1,$ (3.11)

$p_{i,i',j,j_2} \ge \sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i',j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1,$ (3.12)

$p_{i,i',j,j_2} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1,$ (3.13)

$\left(i' - i_1\right)\left(j_2 - j_1 + 1\right) p_{i_1,i',j_1,j_2} \ge \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2} u_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1,$ (3.14)

$q_{i,i,j,j'} \ge \sum_{i=i_1}^{i_2}\sum_{k=1}^{n} z_{i,j',k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1,$ (3.15)

$q_{i,i,j,j'} \ge \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j'-1}\sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1,$ (3.16)

$\left(j' - j_1\right)\left(i_2 - i_1 + 1\right) q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j'-1} v_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > i_1, \forall j' > j_1,$ (3.17)

$u_{i,i,j,j} + \sum_{i'=i_1+1}^{l_2} p_{i,i',j,j} + \sum_{j'=j_1+1}^{j_2} q_{i,i,j,j'} \ge 1 \quad \forall l_1, \forall j_1, \forall l_2 > i_1, \forall j_2 > j_1 \quad ,$ (3.18)

$$vv \varepsilon x + \sum_{j=1}^{n}\sum_{k=1}^{n} vw \angle_{i,j,k} \qquad \forall i, \tag{3.19}$$

$$\text{[illegible]} \qquad \forall j, \tag{3.20}$$

$$z_{i,j,k} \in \{0,1\} \quad \forall i, \forall j, \forall k, \tag{3.21}$$

$$u_{i,j,i'} \in \{0,1\} \quad \forall i, \forall j, \forall i' > i, \tag{3.22}$$

$$v_{j,j,j'} \in \{0,1\} \quad \forall i, \forall j, \forall j' > j, \tag{3.23}$$

$$d_{i_1,i_2,j_1,j_2}, p_{i_1,i_2,j_1,j_2}, q_{i_2,j_1,j_2} \in \{0,1\} \quad \forall i, \forall j, \forall i_2 > i_1, \forall j_2 > j_1, \tag{3.24}$$

En la formulación anterior, las restricciones (3.4), (3.5) y (3.6) aseguran que cada posición horizontal o vertical sea ocupada por exactamente una pieza y cada pieza es empacada exactamente una vez.

Las restricciones [(3.7)-(3.10)] son restricciones para garantizar que no existan traslapes entre piezas. Las restricciones [(3.11)-(3.18)] son restricciones guillotina para cada área rectangular. Si las restricciones guillotina son satisfechas para cada área rectangular, entonces también se cumple para todo el patrón de corte.

Y finalmente las restricciones (3.19) y (3.20) garantizan que ninguna pieza exceda horizontalmente el ancho W y que ninguna pieza exceda verticalmente la altura H, considerando que la función objetivo es maximizar la sumatoria del costo asociado de las piezas empacadas en la placa (ver ecuación (3.1)).

El modelo que representa la versión sin pesos del problema, es igual al anterior, modificado solo en la función objetivo, es decir la ecuación (3.1) es reemplazada por la ecuación (3.25).

$$\text{Maximizar} \sum_{i=1}^{n}\sum_{j=1}^{n} w_k l_k z_{i,j,k} \quad \forall k \tag{3.25}$$

En este modelo existen cerca de $2n^4$ restricciones y cerca de $3n^4/4$ variables binarias (y). Por otra parte, un grupo grande de restricciones presentan un comportamiento donde solo una restricción esta activa y las otras son redundantes (por ejemplo, existe siempre una restricción redundante entre (3.7) y (3.9)).

Por esta razón, la relajación PL del modelo obtiene un límite inferior de mala calidad. El modelo presenta una alta complejidad matemática que en la práctica lo hace inexplotable a través del software y hardware actual disponible para la programación entera mixta (para más detalle de este modelo ver (Ben *et al.*, 2008)).

3.4. Resumen

Los problemas presentados en este trabajo han sido estudiados por más de 6 décadas sin embargo no presentan un modelo matemático bien definido para todas las diferentes características, en especial los patrones de corte guillotina.

En este capítulo se adaptó un modelo propuesto recientemente por (Ben *et al.*, 2008), este describe los patrones de corte tipo guillotina, aunque carece de la restricción de rotación de las piezas 90°. (Lodi *et al.*, 2004) propones variantes para incluir esta restricción.

(Lodi *et al.*, 2004; Ben *et al.*, 2008) concluyen que actualmente no existe software y hardware para darle solución a instancias prácticas del problema mediante los modelos planteados. Por otro lado, intentar a través de soluciones aproximadas puede presentar una metodología novedosas técnicas metaheurísticas de optimización que dé una solución eficiente.

Capítulo 4.

TÉCNICAS DE OPTIMIZACIÓN: OPTIMIZACIÓN CON CÚMULO DE PARTÍCULAS, RECOCIDO SIMULADO Y ALGORITMO GENÉTICO

4.1. Introducción

En esta sección se presentan a grandes rasgos las características más importantes de las tres diferentes técnicas metaheurísticas de optimización. La filosofía, las codificaciones más comunes y el procedimiento general de cada una de las técnicas son presentados a continuación.

4.2. Optimización con cúmulo de partículas (*Particle Swarm Optimización*, PSO)

A principios del siglo XX se da inicio a estudios del comportamiento social de los individuos tratando de entender la influencia que presentaban sobre los demás integrantes de un grupo. Las investigaciones realizadas por (Reynolds, 1994) sobre las aves, simulando su comportamiento individual y colectivo dieron origen a la técnica combinatorial denominada PSO. Basados en estos conceptos (Kennedy y Eberhart, 1995) presentaron la formulación matemática del modelo PSO.

El método PSO es una metaheurística perteneciente al grupo de los algoritmos evolutivos y su estrategia se fundamenta en el comportamiento social de los animales tales como las bandadas de aves, bancos de peces o enjambres de abejas, los cuales actúan como si fueran un solo individuo. En estos grupos de animales se establecen relaciones entre los individuos, se crean jerarquías dependiendo de las características de los mismos, existiendo un líder grupal reconocido y seguido por los demás miembros del grupo. El papel de líder puede cambiar si existe otro individuo con mejores características que el líder existente. Cuando el grupo se organiza para realizar una tarea, el líder guía a los demás individuos hacia una región promisoria, sin embargo, los demás miembros del grupo durante su recorrido pueden inferir sobre una nueva dirección con el objetivo de tener un mejor éxito en la misión.

Esta técnica presenta cierta similitud con los algoritmos genéticos, ya que el proceso incluye una población, realizando una búsqueda óptima a través de actualizaciones de esta. A pesar de no poseer operadores de recombinación y mutación, cuenta con otros operadores que permiten trasladarse a través del espacio de solución y de esta manera detectar soluciones de muy alta calidad, con la posibilidad que una de ellas corresponda al óptimo global del problema.

En PSO la exploración del espacio de solución se realiza a través de una población de individuos conocidos como partículas, donde cada uno de ellas representa una posible solución del problema. La ubicación de cada partícula sobre la región de búsqueda está determinada mediante su posición, la cual, representa el valor que toman las variables de decisión del problema.

Cada partícula cambia de posición de acuerdo a su velocidad teniendo en cuenta la mejor solución encontrada por ésta a lo largo del proceso (*pbest*) y a la información del líder del cúmulo (*gbest*). El operador *pbest* (*individual best*) compara la posición actual

de una partícula con la mejor posición que ha presentado durante el proceso de búsqueda. Mientras que el operador *gbest* (*global best*) estudia el comportamiento del grupo, almacenando la posición del líder actual del cúmulo.

Población inicial

Las técnicas heurísticas dependiendo de la complejidad matemática del problema generan la población inicial de forma aleatoria o utilizando métodos constructivos basados en factores de sensibilidad cuya finalidad consiste en iniciar la búsqueda en una región atractiva con el fin de disminuir tiempo y esfuerzo computacional. El caso de estudio presentado en este trabajo inicia con un conjunto de partículas generadas de forma aleatoria alcanzando el óptimo del problema. Sin embargo, en la medida que se estudian sistemas grandes y de alta complejidad matemática cobra fuerza el uso de población inicial generada con base en métodos constructivos.

Una población está constituida por k partículas y cada una de estas en el estado i representa una alternativa de solución, la cual es interpretada en el problema a través de la posición de la partícula. Cada partícula se representa mediante un vector cuyas posiciones indican los valores de las variables del problema (ver figura 4.1) y simbolizan un punto dentro del espacio de solución. Inicialmente se generan n partículas con valores aleatorios dentro del rango de las variables y a través de la función objetivo se determina la calidad de la solución.

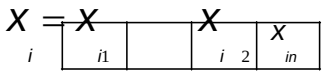

Figura 4.1. Posición de la i-ésima partícula.

Selección del líder

En PSO durante cada iteración se debe seleccionar el líder del grupo comparando los valores de la función objetivo de cada partícula con la función objetivo del líder, siendo éste último aquella partícula que posee el mejor valor (incumbente global). Durante el proceso de optimización, cada vez que una partícula mejore el valor de su función objetivo, se debe actualizar el valor de su mejor ubicación local *pbest* y cada vez que exista un cambio de líder, se debe actualizar el valor de la mejor ubicación global *gbest*.

Función de velocidad

La velocidad permite actualizar la posición de cada partícula. El vector de velocidad representa el gradiente de cada individuo dentro del cúmulo, es decir, guía a las partículas durante el proceso de búsqueda. Al igual que la posición, la velocidad se representa a través de un vector cuyas dimensiones deben ser iguales (ver figura 4.2). La velocidad contiene información de la experiencia local y colectiva del grupo. Para calcular la velocidad de cada partícula se usa la siguiente expresión:

$$v_i(t+1) = w \cdot v_{i-1}(t) + c_1 \cdot rand \cdot \left(pbest - x_i(t) \right) + c_2 \cdot Rand \cdot \left(gbest - x_i(t) \right) \qquad (4.1)$$

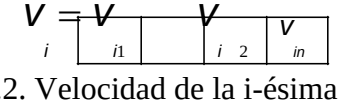

Figura 4.2. Velocidad de la i-ésima partícula.

Donde:

Velocidad actual (v_{i-1}): Dirección de vuelo que presenta una partícula. Al inicio del proceso las partículas parten del reposo.

Factor de inercia (w): Factor de ponderación que actúa sobre la velocidad actual de la partícula. Un valor elevado puede ocasionar una búsqueda muy exhaustiva mientras que un valor demasiado pequeño conlleva a que la exploración se realice de manera fugaz.

Constantes de aceleración c_1 y c_2: Valores que direccionan a las partículas hacia su mejor ubicación local o global respectivamente. La adecuada calibración de estos valores permite que la población no se homogenice durante el proceso.

Rand y *rand*: Valores aleatorios pertenecientes a una distribución de probabilidad uniforme.

Actualización de la posición

Para determinar la nueva posición de las partículas se aplica la siguiente ecuación:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4.2}$$

Como la posición actualizada de las partículas debe satisfacer las restricciones de las variables del problema, es necesario definir un rango de valores para las velocidades evitando de esta manera sobrepasar el límite impuesto.

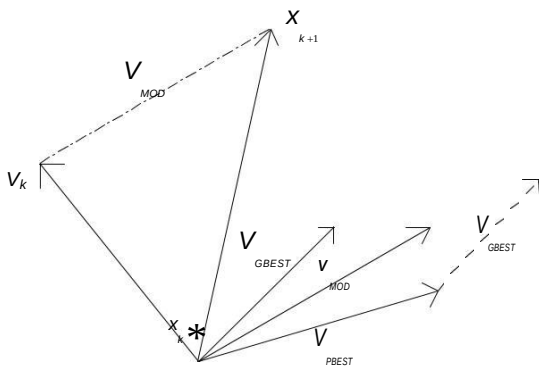

Figura 4.3. Esquema de la nueva posición de la partícula.

Diferentes Codificaciones

Las variables de un problema de optimización pueden ser representadas a través de valores enteros, reales o binarios. Si se usa una representación real no existe problema alguno con los procesos definidos anteriormente, pero si se desea una representación binaria diferentes problemas se encuentran. A continuación se ejemplifica el uso de una codificación real. Sea d_i las variables de decisión del problema ($i = 1, 2, \ldots, n$) y definida formalmente como un conjunto de variables binarias.

$$d_i \in \{0,1\}, \ \forall i$$

Teniendo en cuenta lo anterior, una alternativa de solución dada por una partícula dentro del cúmulo se representa de la siguiente forma:

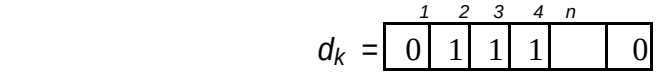

Figura 4.4. Codificación binaria del problema.

En la versión binaria, al igual que en la versión continua, la posición de una partícula es actualizada a través del operador de velocidad (ver ecuación (4.1)), el cual es representado con valores continuos y para este caso se hace necesario implementar una función de activación que permita transformar estos valores a números binarios. La función que se emplea para este propósito se conoce como función sigmoidal (habitualmente utilizada en redes neuronales) cuyo rango de valores se encuentra en el intervalo [0-1] (ver figura 4.5).

$$sig(v_i) = \frac{1}{1+\exp(-v_i)} \tag{4.3}$$

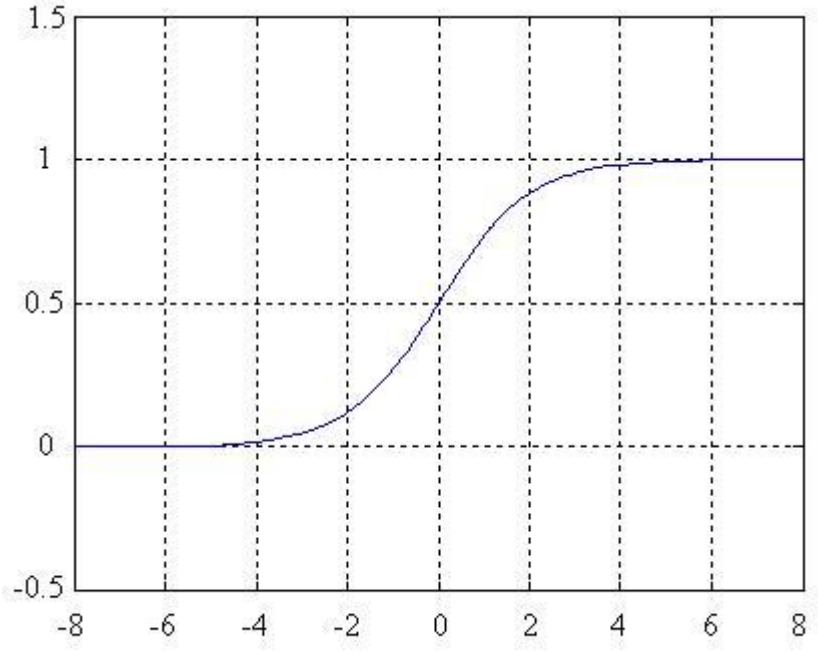

Figura 4.5. Gráfica de la función sigmoidal.

La posición de la partícula asumirá un valor binario (0 ó 1). Dicha posición es actualizada teniendo en cuenta los valores de velocidad, los cuales son calculadas usando la ecuación (4.2). El valor obtenido es comparado con el umbral ρ, el cual puede tomar valores entre [0-1]. El rango típico sugerido es [0.5 - 0.7]. El procedimiento seguido para la actualización de la posición es el siguiente:

$$1, \quad si\ \rho < sig\left(v \right.$$

$$x_{i1} = 0, \quad si\ \rho\ \varepsilon\ sig\left(v_{i1} \right) \tag{4.4}$$

El procedimiento anterior se constituye en una adecuación para el cálculo de la posición de las partículas para la versión binaria de PSO.

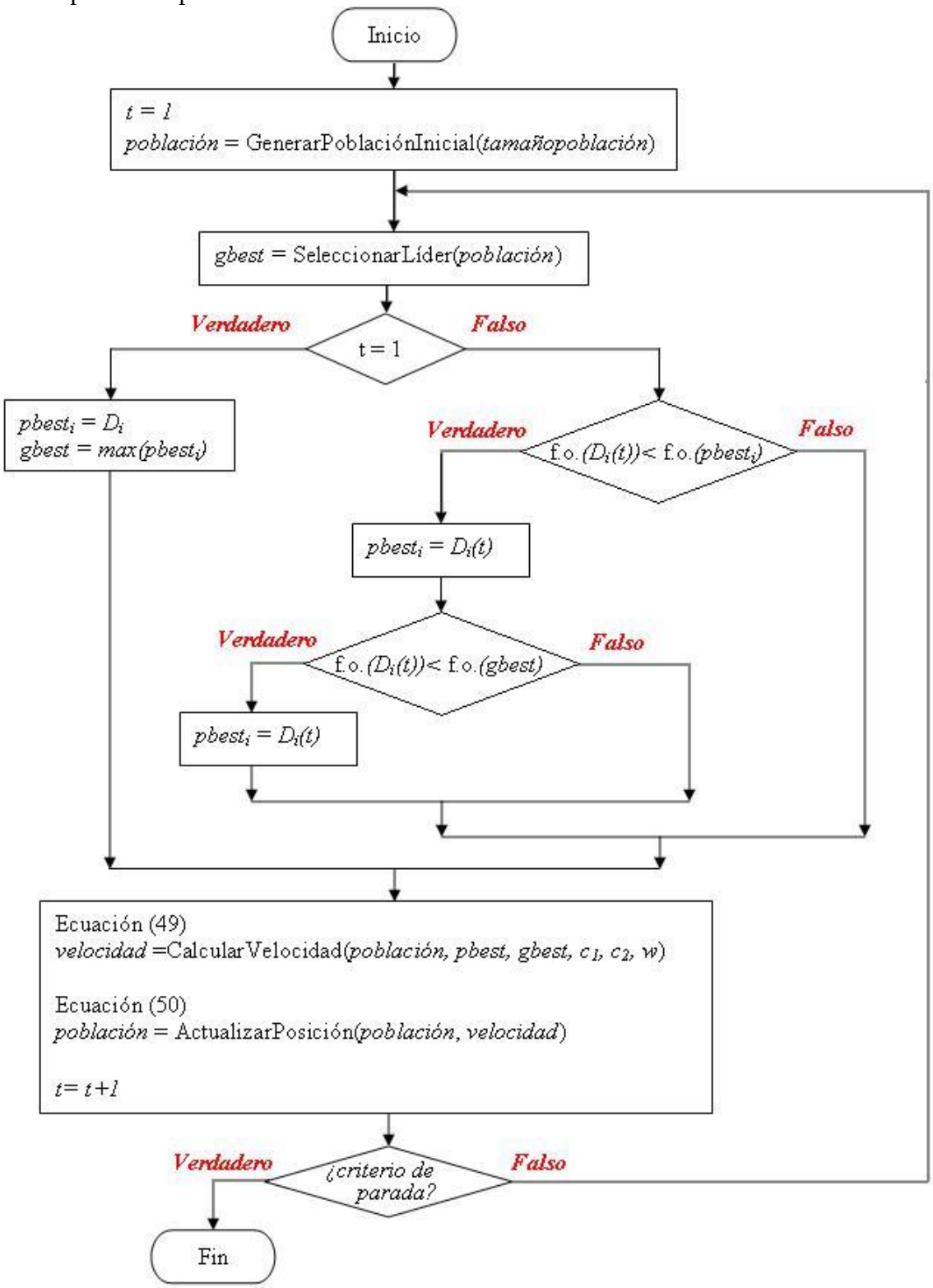

Figura 4.6. Diagrama de flujo de datos del algoritmo PSO.

Criterio de parada

Los algoritmos de optimización precisan de un criterio que les permita decidir cuándo finalizar la exploración del espacio de solución. Entre los criterios de parada más empleados en dichos algoritmos se encuentran:

Si durante un número de iteraciones no se ha mejorado la incumbente, se escoge esta solución y se da por finalizado el proceso.
Hasta cumplir un número máximo de iteraciones.

Siendo la última opción la considerada en este trabajo.

Descripción del algoritmo

El procedimiento empleado por el algoritmo PSO es presentado en la figura 4.6.

4.3. Recocido simulado (*Simulated Annealing*, SA)

El recocido simulado es una técnica metaheurística introducida a comienzos de los 80 por (Kirkpatrick *et al.*, 1983), e inspirada en un procedimiento físico de templado de metales usado en la metalurgia para llevar un sólido a un estado de equilibrio térmico. El recocido consta de tres etapas: la primera consiste en una fase de calentamiento a una temperatura determinada que depende del tipo de material y de su grado de deformación; en la segunda fase se mantiene la temperatura, permitiendo de esta manera, una reorganización molecular en estados de mínima energía. Finalmente se realiza la fase de enfriamiento controlado, disminuyendo gradualmente la temperatura, hasta lograr un sólido cuyas partículas se encuentran en un estado cristalino perfecto. Al iniciar la etapa de enfriamiento, para cada valor de temperatura debe permitirse el alcance del equilibrio térmico. De lo contrario el sólido presentará una estructura amorfa en lugar de la estructura cristalina de más baja energía.

El procedimiento que dio origen a esta metodología de optimización se denomina algoritmo de Metropolis propuesto por (Metropolis *et al.*, 1953), con el que se estudian las propiedades de equilibrio en el análisis del comportamiento microscópico de los cuerpos. El algoritmo de Metrópolis se basa en técnicas de simulación de Monte Carlo generando una secuencia de estados de un sólido, es decir, dado un sólido en un estado i y con energía E_i, se genera el siguiente estado j con energía E_j, mediante la aplicación de un mecanismo que lo conduce al estado siguiente a través de una pequeña perturbación. Si la diferencia de energía $E_j - E_i$ es menor o igual a cero, el estado j es aceptado. Si la diferencia de energía es mayor que cero, el estado j es aceptado con cierta probabilidad, la cual está dada por:

$$P_{aceptacion} = \exp^{\frac{E_j - E_i}{k_b T}} \tag{4.5}$$

Donde T representa la temperatura y k_b es la constante de Boltzmann. Esta regla de aceptación se conoce como *criterio de Metrópolis*.

De forma análoga el criterio de aceptación empleado en el algoritmo SA determina si *j* se acepta a partir de *i* al comparar los valores de la función objetivo (*f*) y aplicando la ecuación (4.6) para calcular la probabilidad de aceptación.

$$r_{aceptacion} = \begin{cases} 1, & si\ f(j) \leq f(i) \\ \exp\left(\frac{f(i)-f(j)}{T}\right), & si\ f(j) > f(i) \end{cases} \quad (4.6)$$

En la ecuación (4.6) ilustra la probabilidad de aceptar una nueva configuración de peor calidad, resulta de comparar la expresión correspondiente con un número aleatorio generado dentro de una distribución de probabilidad uniforme en el intervalo [0,1].

Los parámetros que intervienen en el algoritmo son presentados a continuación:

Temperatura inicial

El valor inicial de temperatura T_0 se obtiene de forma constructiva, simulando las transiciones hechas al inicio del proceso para la primera cadena de Markov.

El concepto físico de temperatura dentro del algoritmo SA no presenta un significado real, sino que ha de ser considerado como un parámetro por calibrar. Existen diversas formas de calcular la temperatura inicial, una de las más utilizadas se describe a continuación:

Cálculo temperatura inicial	
1.	Generar una alternativa de solución.
2.	Inicializar $T_o = 0$.
3.	Ejecutar la cadena m_o .
4.	Crear una nueva alternativa.
5.	Si $(f(i) - f(j)) \leq 0$ entonces
5.a.	$m_1 = m_1 + 1$ y aplique la fórmula (9) para el cálculo de T_o De lo contrario
5.b.	$m_2 = m_2 + 1$ y aplique la fórmula (9).
6.	Si $m_0 = m_1 + m_2$ entonces
6.a	Terminó la cadena y el valor de T_o (calculado en la última iteración) se asume como el valor inicial de temperatura.

Tabla 4.1. Algoritmo para calcular la temperatura inicial.

$$T_0 = \frac{F^+}{\ln\left(\frac{m_2}{m_2 X - m_1 (1 - X)}\right)} \quad (4.7)$$

Donde la ecuación (4.7) describe el cálculo de T_0, m_1 es el número de transiciones propuestas de *i* a *j* para las cuales $f(i) \leq f(j)$, m_2 es el número de transiciones propuestas de i a j para las cuales *f(i)>f(j)* y F^+ es el incremento medio en el costo de las m_2 transiciones, calculado mediante la ecuación (4.8).

$$F+ = \frac{\sum_{i=1}^{m_2} F_i}{m_2} \tag{4.8}$$

Longitud de la cadena

A medida que la temperatura baja, también lo hace la probabilidad de aceptación y es necesario aumentar el número de posibilidades a evaluar. Este número de posibilidades en cada paso del proceso iterativo se conoce con el nombre de cadena de Markov o número de tentativas N_k. La longitud de la cadena N_k define el número de vecinos por explorar en el nivel k de temperatura T_k. Como regla general se maneja una longitud creciente a medida que la temperatura disminuye, con el fin de alcanzar el equilibrio de energía y minimizarla para cada estado de temperatura.

La expresión utilizada para actualizar el valor de la longitud de la cadena se presenta a continuación:

$$N_{k+1} = \rho \cdot N_k \tag{4.9}$$

Donde ρ es un valor fijo para el incremento de la cadena y es recomendado en la literatura especializada en un rango de [0,01 – 0,10].

Tasa de enfriamiento

Corresponde al porcentaje de disminución de la temperatura, afectando la probabilidad de aceptación de soluciones de peor calidad. El cálculo de la temperatura en el nuevo ciclo iterativo puede efectuarse de la siguiente manera:

$$T_{k+1} = \eta \cdot T_k \tag{4.10}$$

El valor de η es recomendado en la literatura especializada en un rango de [0,8 – 0,99].

Definición de vecindad

La estructura de vecindad es determinante para el buen desempeño del algoritmo, consiste en definir la perturbación necesaria para pasar de un estado a otro. Esta depende de la codificación propuesta y el problema que se está trabajando. Para ejemplificar la definición de vecindad para una codificación binaria, se tiene el siguiente esquema:

Integrar (hacer uno) una variable de decisión en el vector solución.
Retirar (hacer cero) una variable de decisión en el vector solución.
Intercambiar los valores de dos variables en el vector solución.
Ingresar (hacer uno) una variable y retirar (hacer cero) otra variable.

Todas las alternativas cuentan con igual probabilidad de ocurrencia o inicialmente intente el ingreso; de ser aceptada regrese a otra propuesta de vecindad, en caso contrario, intenta retirar.

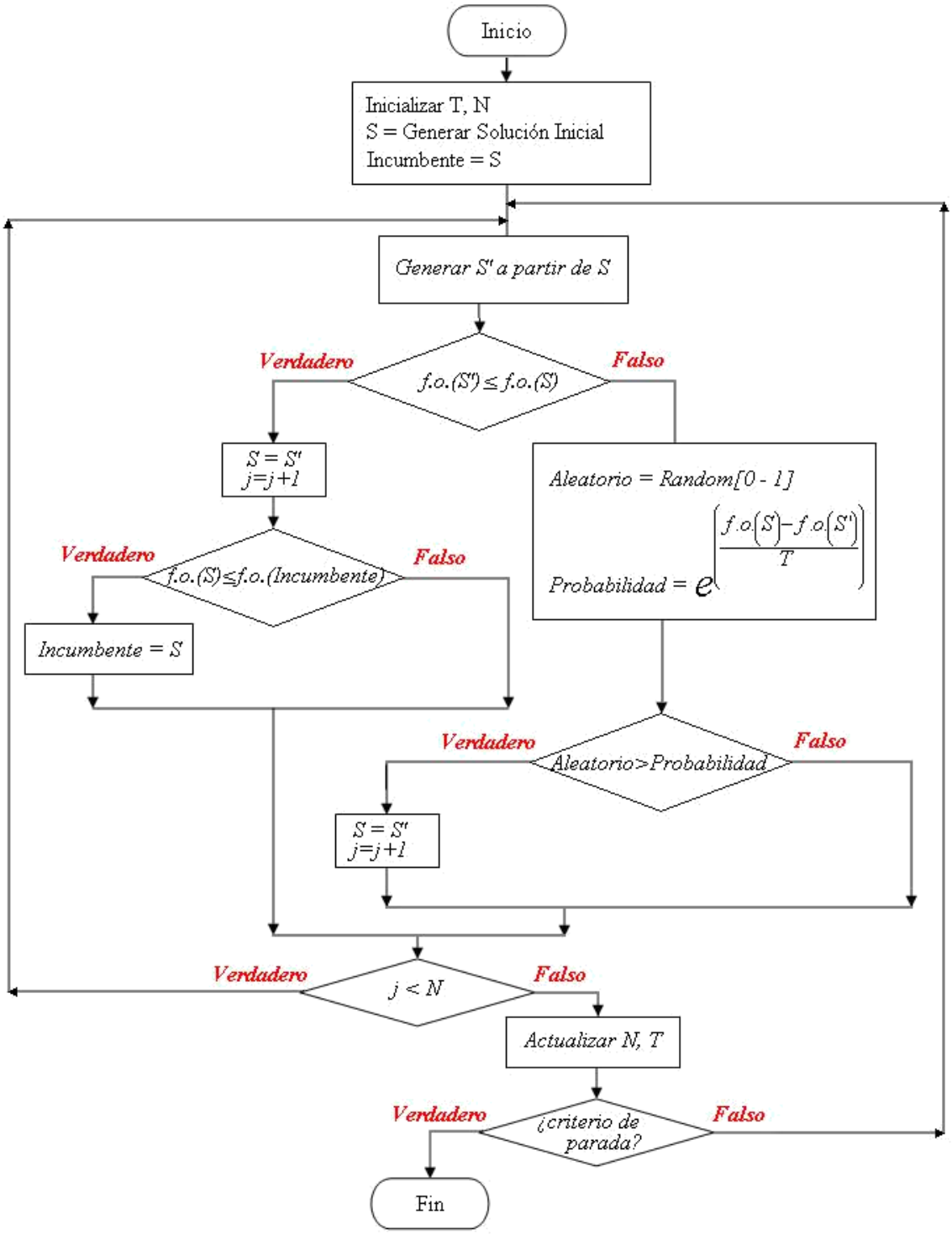

Figura 4.7. Diagrama de flujo de datos del algoritmo SA

Criterio de parada

Existen diferentes procedimientos para definir el criterio de parada:

Fijar un número determinado de niveles de temperatura a analizar.

Detener la búsqueda al finalizar el nivel de temperatura T_k si en este no se obtiene una solución de mejor calidad que la encontrada en los niveles anteriores.

Detener la búsqueda cuando en el último nivel de temperatura no se haya encontrado un número mínimo de aceptaciones.

Ejecutar el proceso un número especificado de iteraciones.

Descripción del algoritmo SA

El procedimiento empleado por el algoritmo SA es presentado en la figura 4.7.

4.4. Algoritmo genético (*Genetic Algorithm*, GA)

Historia

El origen de los Algoritmos Genéticos se dio a finales de los años 50 con la realización de trabajos y publicaciones en el campo de los algoritmos evolutivos como alternativa de solución para problemas combinatorios. Esta técnica se define como una familia de modelos computacionales inspirados en los mecanismos de evolución natural perteneciente a la familia de los algoritmos evolutivos tomados de la teoría de (Darwin, 1859) sobre la evolución de las especies.

Filosofía

Los GA emplean un conjunto de soluciones en cada iteración del algoritmo en lugar de emplear una única solución como las metaheurísticas basadas en trayectorias. Mediante una imitación del mecanismo genético de los organismos biológicos, los algoritmos genéticos exploran el espacio de soluciones para un problema determinado. Una alternativa de solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos alelos. Los cromosomas son sometidos a un proceso de evolución que abarca la evaluación, selección, recombinación y mutación. Después de varios ciclos de evolución la población deberá contener individuos más aptos.

Un algoritmo genético elemental realiza la siguiente secuencia de operaciones:

1. Genera una población inicial después de escoger el tipo de codificación para representar cada configuración.
2. Calcula la función objetivo de cada configuración de la población y almacena la incumbente.
3. Realiza selección.
4. Realiza recombinación.
5. Realiza mutación y termina de generar la nueva población de la siguiente generación.
6. Si el criterio de parada (o criterios de parada) no se han cumplido el proceso regresa al paso 2.

Los pasos (2), (3), (4) y (5), en conjunto, son conocidos como ciclo generacional. También es necesario mencionar que existe una equivalencia entre los términos usados en genética y en un problema de optimización matemática.

Selección

Este operador genético permite seleccionar las configuraciones de la población actual que deben participar en la generación de las configuraciones de la nueva población (nueva generación). Por tanto la función del operador de selección termina después de decidir el número de descendientes que debe tener cada configuración de la población actual. En algunos casos las configuraciones puede generar varios descendientes mientras que en otras ninguno, desapareciendo la información de estas configuraciones que son consideradas de baja calidad.

Selección proporcional

La forma más simple de implementar la selección es usando el denominado esquema de selección proporcional. En esta estrategia cada configuración tiene derecho a generar un número de descendientes que es proporcional al valor de la función de adaptación. Así se tiene la siguiente relación.

$$Nd_i = \frac{z_i(X)}{z_m(X)} \tag{4.11}$$

Nd_i = Número de descendientes de la configuración i.
n = número de configuraciones de la población. $z_i(X)$ = Función de adaptación.
$z_m(X)$ = Media de la función objetivo

$$z_m(X) = \frac{1}{n}\sum_{i=1}^{n} z_i(X) \tag{4.12}$$

$$Nd_i = n\frac{z_i}{\sum_{i=1}^{n} z_i(X)} \tag{4.13}$$

La propuesta inicial y aún usada para resolver el problema de número de descendientes no entero es el denominado esquema de selección de la ruleta. En el esquema de selección de la ruleta a cada configuración se le asigna una parte de la ruleta que es proporcional al número de descendientes, calculada con la siguiente relación:

$$2\pi \frac{1}{n} Nd_i \tag{4.14}$$

Selección por torneo

La selección por torneo, es bastante simple y consiste en escoger la mejor alternativa de las k alternativas seleccionadas aleatoriamente. El éxito de aplicar esta metodología radica en escoger adecuadamente un valor de k que se ajuste a cada problema en particular teniendo en cuenta tamaño y complejidad del problema, así como el tamaño de la población inicial. Un valor de k muy alto y la generación de una población inicial

pequeña pueden, eventualmente, hacer elitista el proceso de selección y ocasionar convergencias locales. Un valor muy bajo de *k* puede ocasionar un mayor esfuerzo computacional (Gallego *et al.*, 2008).

Recombinación (*Crossover*)

Las configuraciones seleccionadas en el proceso de selección son sometidas a recombinación. Este operador consiste en intercambiar partes de dos vectores para formar dos nuevos vectores donde uno de los vectores nuevos tiene parte de los elementos de un vector y parte de los elementos de otro vector. Este operador se denomina también cruzamiento e intenta simular el fenómeno del *crossing over* en genética. Generalmente a las configuraciones seleccionadas (originales) se les denomina configuraciones padres y a las nuevas configuraciones se les denomina configuraciones hijos.

Recombinación de Punto Simple

Es la forma más simple de recombinación y consiste en escoger un único punto para realizar la recombinación. Para explicar el concepto, se supone que una configuración tiene k elementos o celdas binarias; una vez elegidas las dos configuraciones para realizar recombinación, se genera un número aleatorio entre 1 y (*k*-1), y ese número indica el punto de recombinación. La parte que se encuentra hacia la derecha de ambas configuraciones son intercambiadas para formar las dos nuevas configuraciones. Las configuraciones que deben ser sometidas a recombinación son escogidas aleatoriamente entre las configuraciones seleccionadas que todavía tienen derecho a recombinación. Aquí están nuevamente presentes dos nuevas decisiones de carácter aleatorio: (1) Se escoge de forma aleatoria, las configuraciones que deben ser sometidas a recombinación y (2) Se escoge aleatoriamente el punto de recombinación. Aunque algunas decisiones determinísticas son usadas en algunos casos.

Algunos problemas que se presentan con este operador son:

El mismo par de configuraciones pueden ser escogidas varias veces, especialmente cuando existen configuraciones que tienen derecho a generar muchos descendientes.

Una misma configuración puede ser escogida para implementar recombinación con ella misma. Un mecanismo simple elimina este problema escogiendo otra configuración cada vez sucede este problema.

Para eliminar estos dos problemas se puede escoger una estrategia más determinista. Por ejemplo, se puede iniciar la recombinación escogiendo primero aquella configuración que tiene derecho a generar un mayor número de descendientes hasta agotar todas las recombinaciones a que esta configuración tiene derecho, después se escoge la segunda en prioridad y así sucesivamente.

Mutación

En la codificación binaria la mutación significa simplemente cambiar el valor de una variable de 0 para 1 ó viceversa. En los trabajos teóricos iniciales de algoritmos

genéticos la mutación siempre se consideró un operador secundario, hoy en día se está reevaluando este concepto.

La tasa de mutación ρ_m indica la probabilidad de que una posición (celda binaria) puede tener su valor actual modificado. En el análisis teórico, y en las propuestas originales, se sugiere que la mutación debe ser intentada bit por bit (celda por celda) y así la decisión de mutación de una posición binaria es independiente de la mutación realizada de otras celdas binarias de una configuración.

Ciclo Generacional

Es el conjunto de procesos de selección, recombinación y mutación que permiten encontrar las configuraciones de la nueva generación (población) a partir de la población actual. El ciclo generacional es controlado por el programa de control del algoritmo genético.

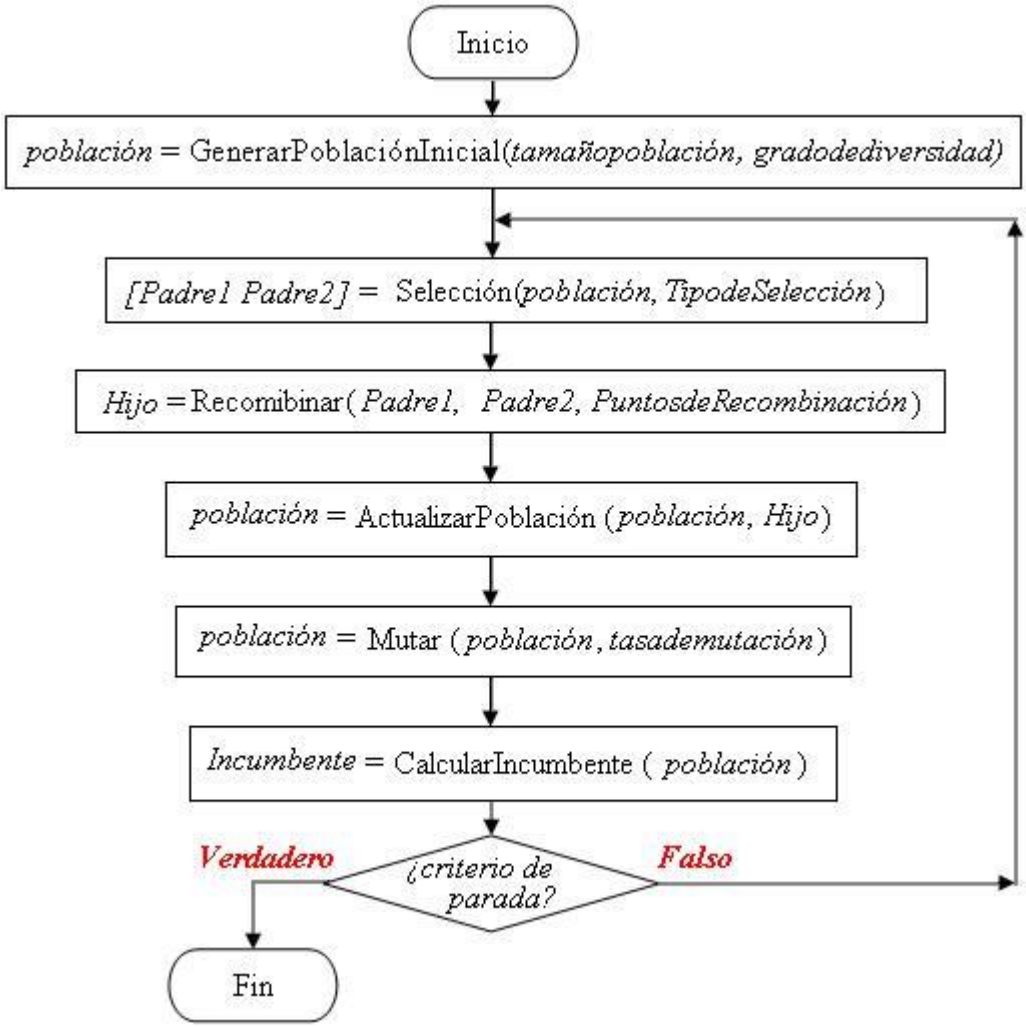

Figura 4.8. Diagrama de flujo de datos del algoritmo GA básico.

Criterio de Parada

Existen varios criterios de parada.

- Se ha realizado un número específico de generaciones.

- La incumbente alcanza un valor de una calidad mínima especificada.
- La población es demasiada homogénea, es decir, las configuraciones son similares y no existe más evolución.

En implementaciones prácticas de problemas complejos se especifican criterios de parada más objetivos y generalmente se especifica más de un criterio de parada en un mismo algoritmo.

Descripción del algoritmo básico

El procedimiento empleado por el algoritmo GA básico es presentado en la figura 4.8.

Algoritmo genético de Chu-Beasley

El algoritmo genético de Chu-Beasley es una versión modificada del GA básico con ciertas características que lo convierten en un algoritmo más eficiente. La principal característica de este algoritmo consiste en mantener diversidad entre los cromosomas que conforman la población durante todo el proceso. En cada generación es reemplazado un solo cromosoma (alternativa) en la población, siempre y cuando, cumpla con las condiciones de optimalidad y/o factibilidad establecidas. Dicho mecanismo busca que en cada ciclo generacional, la calidad de la solución sea mejorada por optimalidad y/o factibilidad. Durante el proceso en la población se reemplaza sistemáticamente un único descendiente. Esta estrategia tiene como ventaja encontrar soluciones de alta calidad y garantizar diversidad en la población a lo largo de las generaciones.

En el GA de Chu-Beasley el operador de selección escoge un número K de padres (generalmente se asume $K=2$). A estas configuraciones se les aplica el operador de recombinación dando como resultado K descendientes, de los cuales se escoge una sola configuración de manera aleatoria para continuar con el operador de mutación. Terminado el proceso generacional se tiene un único descendiente (configuración actual), el cual debe cumplir con las siguientes estrategias de optimalidad y/o factibilidad para formar parte de la población inicial:

> Si la alternativa actual es infactible y a su vez es menos infactible que la peor infactible de la población, entonces reemplazar la peor infactible por la alternativa actual.
>
> Si la configuración es factible y existe por lo menos una infactible en la población actual, entonces reemplazar la peor infactible por la alternativa actual.
>
> Si la configuración es factible y todas las alternativas de la población actual son factibles, entonces reemplazar la alternativa con peor función objetivo por la alternativa actual. Lo anterior se realiza sólo si la alternativa actual es de mejor calidad que la peor de la población.

En caso contrario desechar la alternativa resultante e iniciar un nuevo ciclo generacional.

4.5. Resumen

Se presentó para las técnicas metaheurísticas: cúmulo de partículas, algoritmo genético y recocido simulado, las principales características, su filosofía, sus diferentes codificaciones y su procedimiento general.

En general las técnicas: algoritmo genético y recocido simulado son altamente reconocidas por sus excelentes resultados en diferentes problemas clásicos de optimización. Por otro lado, la técnica de cúmulo de partículas es relativamente nueva, pero estudios recientes han demostrado su gran desempeño y sus buenos resultados.

Capítulo 5.

CODIFICACIÓN DEL PROBLEMA Y ADAPTACIÓN DE LAS TÉCNICAS DE OPTIMIZACIÓN COMBINATORIA

5.1. Introducción

La solución de problemas combinatoriales, normalmente está asociada a un proceso de búsqueda. Esta no puede realizarse en forma exhaustiva, por lo que los investigadores han desarrollado una gran cantidad de métodos alternativos de búsqueda que encuentran soluciones de "buena calidad" y en muchos casos inclusive óptimas.

La manera empírica de solucionar problemas combinatorios es a través del ensayo y el error. Enumerando todas las combinaciones posibles, de las cuales se seleccionan las soluciones que satisfagan las condiciones del problema planteado. En la mayoría de los casos, "generar y probar" no es aceptable en términos del espacio y tiempo computacional. Esto es obvio si el sistema de combinaciones es infinito. Pero aunque sea finito muchas veces es muy grande (el espacio de combinaciones crece de forma exponencial). En este caso, la generación de combinaciones genera una explosión combinatoria en el cuál sólo es posible realizar una búsqueda exhaustiva invirtiendo varios miles de millones de años.

5.2. Heurísticas en optimización combinatoria

El uso de métodos heurísticos se fundamenta en los problemas de optimización pertenecientes a la categoría denominada NP (no existe un algoritmo de solución de tiempo polinomial). Si se demuestra que un problema de optimización pertenece a esta categoría, es normal solucionarlo por medio de heurísticos de optimización. Problemas donde su espacio de búsqueda aumenta exponencialmente con el tamaño del problema, es lógica la utilización de algoritmos heurísticos, el problema de empaquetamiento presentado en este trabajo hace parte de esta categoría.

La gran ventaja del uso de heurísticas es la flexibilidad en el manejo del problema a diferencia de las rigurosas técnicas clásicas de la investigación operativa. La mayor desventaja de los métodos heurísticos es que se desconoce la calidad de la solución obtenida, por lo tanto, no se puede estimar la cercanía de dicha solución con respecto al óptimo global.

El uso de métodos heurísticos es apropiado bajo una o más de las siguientes condiciones del problema:

- No existe un método exacto de solución, o en el caso de que dicho método exacto exista, el mismo requiere de mucho gasto computacional o de memoria.
- No es necesario encontrar la solución óptima, en el sentido del óptimo global sino que es suficiente con obtener una solución suficientemente buena.
- Los datos son poco confiables y por tanto no tiene sentido el tratar de encontrar el óptimo global para dichos datos, ya que el mismo no puede ser más que una aproximación al óptimo global que corresponde a los datos correctos.

- Existen limitaciones de tiempo en proporcionar la respuesta o de memoria en computador que va a efectuar los cálculos
- Se va a utilizar el resultado proporcionado por el heurístico de optimización como solución inicial para un algoritmo exacto de tipo iterativo, el cual reduciría considerablemente el número de iteraciones si parte de una solución inicial suficientemente buena.

Se distinguen cuatro tipos de búsquedas heurísticas: constructivas, mejora de una solución, de descomposición y de reducción.

Las búsquedas heurísticas constructivas consisten en ir añadiendo componentes individuales a la solución inicial hasta que se obtiene una solución inicial factible.

Las búsquedas heurísticas basadas en la mejora de una solución, en cada paso se parte de una solución para buscar en su vecindad una solución mejor que la reemplace.

Las búsquedas heurísticas basadas en descomposición dividen el problema en subproblemas más manejables, de modo que al resolver dichos subproblemas se obtiene una solución al problema inicial por integración de las soluciones obtenidas en cada subproblema.

Finalmente, las búsquedas heurísticas basadas en la técnica de reducción. Se trata en este caso de identificar alguna característica que presumiblemente debe poseer la solución óptima, para de este modo simplificar el problema de búsqueda.

Las heurísticas pueden ser dependientes del problema o independientes del mismo. Las primeras, conocidas como heurísticas, son válidas únicamente para el problema particular para el que han sido diseñadas, mientras que las segundas, las llamadas metaheurísticas, pueden aplicarse a cualquier problema.

5.3. Heurísticas constructivas de la mochila bidimensional irrestricta.

Bottom Left (BL) y *Bottom Left Fill* (BLF)

El BL determina que cada pieza a colocar se posiciona inicialmente en la esquina superior derecha de la placa. A continuación, se desplaza hasta la posición más profunda posible. Luego, la pieza es movida hacia la izquierda tanto como sea posible, repitiéndose esta rutina hasta que la pieza alcance una posición inamovible.

El número máximo de patrones posibles de empaquetamiento usando BL son $2^n \cdot n!$ (donde n es el número de piezas). El valor anterior no considera la posibilidad de rotar las piezas, permitir esta variante aumenta considerablemente el número de patrones existentes. El problema fundamental que presenta esta estrategia es que conduce a soluciones con gran número de desperdicios o residuos en la placa, además que no genera todos los patrones de corte existentes también genera patrones de corte no guillotina. La figura 5.1 ilustra el proceso de ubicación de las piezas mediante el BL.

El BLF consiste en encontrar un espacio para la pieza comenzando desde la esquina inferior izquierda de la placa, buscando nivel a nivel desde el fondo de la placa. En caso

de no encontrar ubicación para la pieza dentro de un determinado nivel se pasa al nivel inmediatamente superior. Repitiendo este proceso para todas las piezas.

El BLF intenta minimizar los desperdicios de espacio del BL, diferenciándose del BL, en que antes de colocar la pieza se comprueba que ésta no cabe en ninguno de los espacios generados hasta el momento a un nivel inferior. La figura 5.1 ilustra el proceso de ubicación de las mediante el BLF.

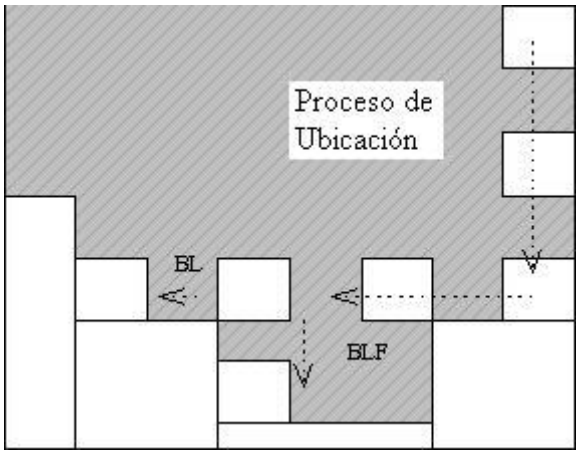

Figura 5.1. Ubicación de las piezas mediante BL y BF

Difference Process (DP)

El DP intenta colocar cada pieza en la esquina más cercana a la esquina inferior izquierda de la placa. Para medir cuál es la posición se emplea la distancia euclidiana. Al igual que con los métodos anteriores existen patrones que no puede alcanzarse con esta estrategia y también genera patrones no guillotina.

La figura 5.2 ilustra las diferencias entre los patrones BL y DP para la ubicación de 5 piezas.

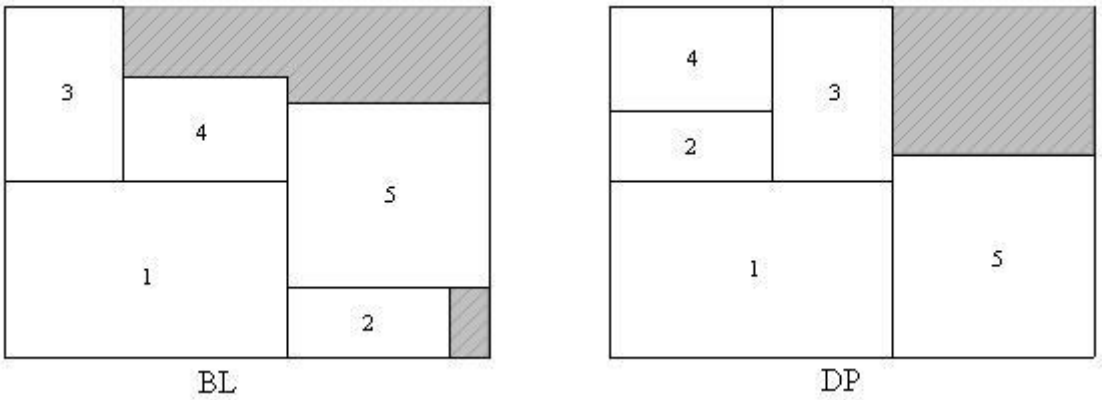

Figura 5.2. Diferencias entre los patrones BL y DP.

BL modificado para corte guillotina

El algoritmo del BL modificado consiste en:

a) Ordenar las piezas por sus alturas en orden descendente.

b) Ubicar la pieza más alta en la parte inferior izquierda, teniendo en cuenta que la altura de esta pieza debe ser menor ó igual que la altura del tablero. El ancho de la pieza insertada indica la primera coordenada guía en el eje *x*.

c) La siguiente pieza es la más alta entre las restantes del proceso; el ancho de esta pieza es evaluado y si sumado a la coordenada guía en el eje *x*, si el valor obtenido es menor que el ancho del tablero, se posiciona al lado derecho de la última pieza insertada en las coordenada guías en los ejes *x* e *y*. Este procedimiento se repite hasta ubicar la mayor cantidad de piezas permitidas por el ancho del tablero, conformando así el nivel actual. La nueva coordenada guía en el eje *x* es la acumulación de los anchos de las piezas del presente nivel.

d) Se evalúa la diferencia de alturas de piezas consecutivas, con el fin de identificar espacios donde se pueden encajar piezas respetando el límite superior impuesto por la pieza más alta del nivel actual.

e) Cuando ya no es posible ubicar más piezas en un nivel el proceso descrito anteriormente es iniciado de nuevo a partir del paso *c*. La nueva coordenada guía en el eje *y* es la acumulación de alturas de las piezas más altas de los niveles anteriores y la nueva coordenada guía en el eje *x* es 0.

f) El proceso para cuando no es posible ubicar más piezas sin sobrepasar la altura del tablero.

5.4. Codificaciones de la mochila bidimensional irrestricta

El uso de metaheurísticas requiere de una codificación apropiada del problema, existen distintas codificaciones propuestas en la literatura especializada, dos de las más usadas son la representación mediante estructura de datos tipo vector y la tipo árbol de cortes.

La representación natural de un patrón de empaquetamiento se basa en las coordenadas de ubicación de cada rectángulo en la placa. Si la esquina inferior izquierda y la esquina superior derecha de todos los rectángulos son conocidas, entonces el patrón de empaquetamiento puede ser reconstruido fácilmente. La ventaja de una representación natural está ligada con una fácil reconstrucción. Pero si hay cambios en las coordenadas es probable que el patrón de empaquetamiento se traslape.

Por ejemplo, (Jakobs, 1996) hace uso de la estructura de datos tipo vector para codificar los patrones de empaquetamiento de su algoritmo genético, por otro lado, (Kröger, 1995) utiliza la estructura de datos tipo árbol de cortes para codificar los patrones de empaquetamiento.

Codificación por permutaciones para corte no guillotina propuesta por Jakobs

Un patrón de empaquetamiento puede ser representado por un permutación $\pi = (i_1,..., i_n)$, (donde, i_j indica que la pieza i ocupa la posición j del vector). La permutación representa la secuencia en la cual los rectángulos son empacados. La ventaja de este tipo de estructura es la fácil creación de nuevas permutaciones cambiando la secuencia. Una consecuencia de esta estructura es que en cada permutación es asignado un único patrón de empaquetamiento. Para decodificar el genotipo es necesario un algoritmo eficiente que lo convierta en la representación natural.

Una dificultad encontrada es cuando dos permutaciones representan el mismo patrón de empaquetamiento (ver figura 5.3). Esta es difícil de subsanar.

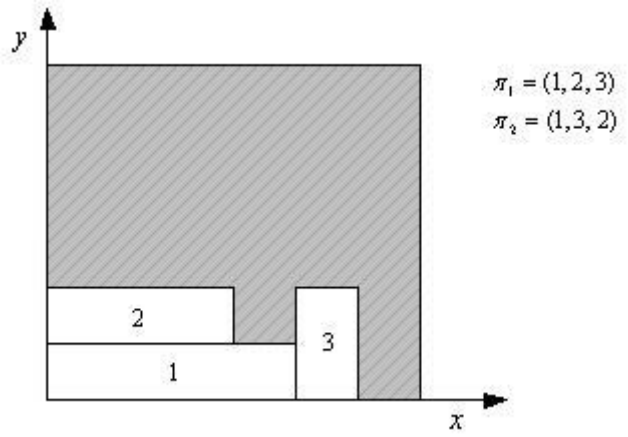

Figura 5.3. Dos permutaciones con el mismo patrón de empaquetamiento.

Codificación a través de dos vectores para corte guillotina

Una alternativa de solución π está codificada por dos vectores. El primero, denominado piezas, contiene una secuencia de ubicación de piezas según el tipo. Dicho vector es de un tamaño igual al número de piezas a ubicar. El segundo vector se denomina secciones y contiene los intervalos para los cuales el primer vector ubica las piezas en una misma sección. El número total de secciones de cada alternativa es determinado de forma aleatoria. Por lo anterior, una alternativa π es un arreglo de dos vectores a los cuales se puede hacer referencia como π piezas y π secciones. La altura de una sección está determinada por la placa de mayor altura.

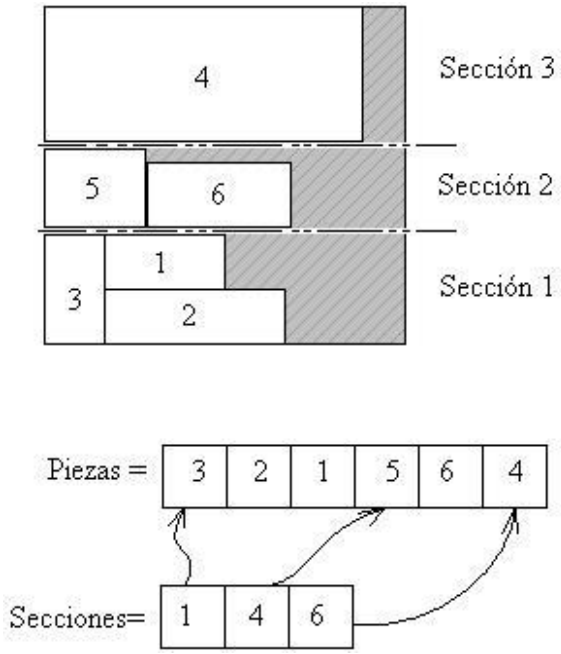

Figura 5.4. Ubicación de las piezas para los vectores: piezas y secciones.

Una configuración factible es aquella donde se ubiquen algunas piezas de las demandadas de forma que se minimice el área sin utilizar. La secuencia de llenado de

las piezas siempre se realiza de izquierda a derecha en cada sección. La figura 5.4 ilustra la ubicación para los vectores de piezas y secciones ejemplo.

Codificación en diagrama de árbol para corte guillotina

La creación de la estructura de árbol de cortes se realiza aplicando técnicas numéricas de conglomerados (clusters) tiene por objeto agrupar elementos. Cada nodo interno del árbol representa la forma en que se realiza el corte y los elementos que pertenecen a cada grupo (Wong *et al.*, 1988).

Se divide la placa original en subespacios. Para asegurar que los subespacios creados presenten cortes de tipo guillotina se define una codificación de árbol binario complementario (Toro *et al.*, 2008).

El número de capas (c) es fijado y determina el número de variables en la codificación así como el número de subespacios (S) creados. El número de cortes (p) está dado por la ecuación 5.1, mientras el número de subespacios (N_S) está dado por la ecuación 5.2.

$$p = 2^c - 1 \tag{5.1}$$

$$N_S = 2^c \tag{5.2}$$

Para representar la estructura que genera el subespacio son definidos dos vectores de tamaño igual al número de cortes: el primero ($\vec{T}$) es un vector de tipo binario que define el tipo de corte (vertical u horizontal), el segundo ($\vec{H}$) es un vector real con valores entre 0 y 1 que determina la distancia porcentual a la cual se produce el corte con respecto al tablero disponible.

Por ejemplo, la figura 5.5 ilustra un árbol de cortes fijando el número de capas en 2. Una ventaja de este tipo de codificación es que cualquier conjunto $\{\vec{T}, \vec{H}\}$ es factible siempre y cuando $\vec{T}$ sea binario y $\vec{H}$ este en el intervalo [0, 1].

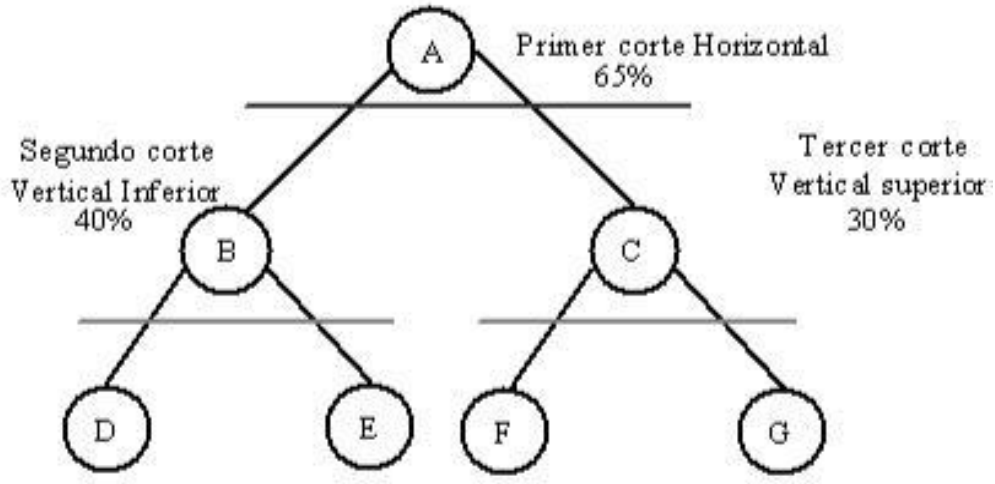

Figura 5.5. Árbol de cortes

La figura 5.6 ilustra la disposición de los cortes sobre la placa del árbol de cortes de la figura 5.5.

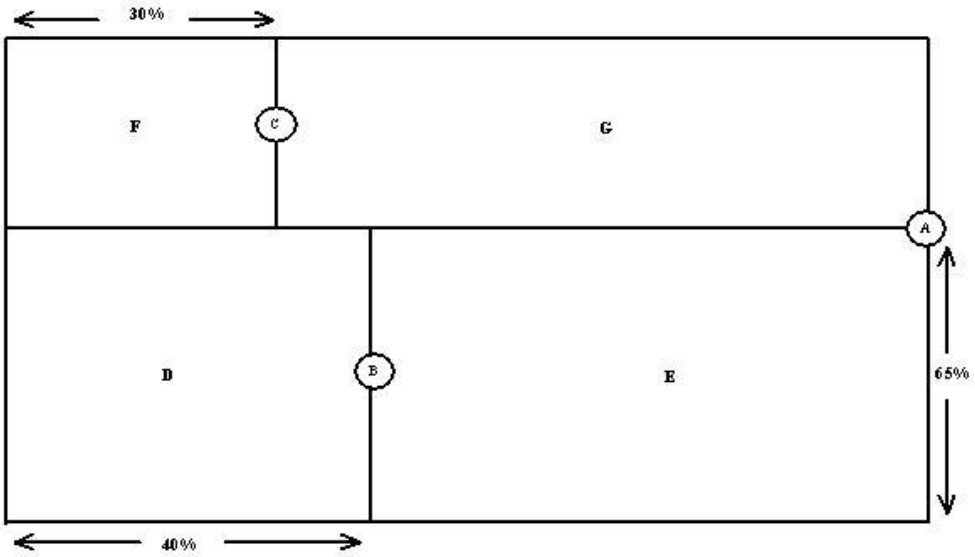

Figura 5.6. Cortes sobre la placa dado por el árbol de corte de la figura 5.5.

En cada uno de los cortes propuestos se acomoda la mayor cantidad del mismo tipo de piezas disponibles de manera que se garantice el corte guillotina, cada uno de los subespacios se llena de acuerdo a un constructivo que calcula el desperdicio o mayor beneficio para la versión con pesos. En la figura 5.7 se presenta una configuración obtenida aplicando la técnica propuesta.

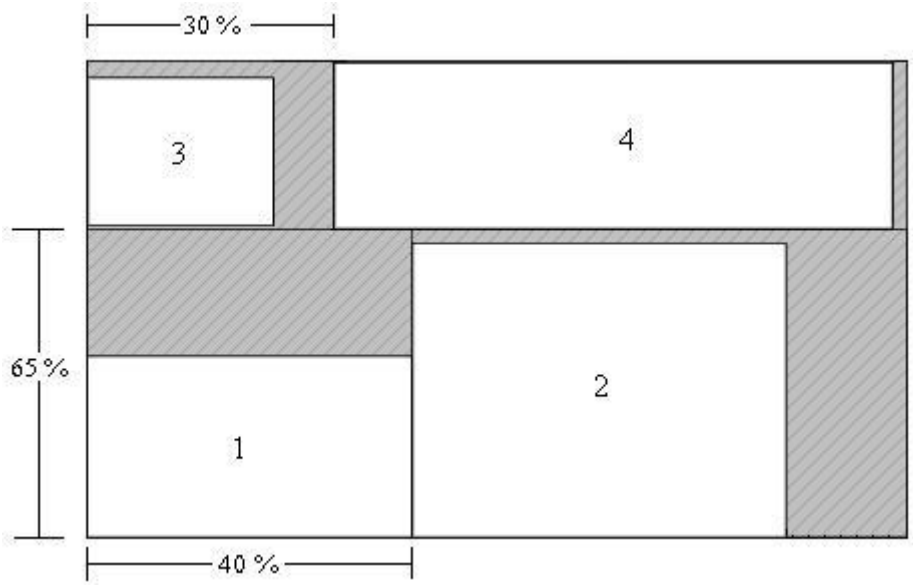

Figura 5.7. Disposición de piezas sobre los subespacios.

Codificación Rectilinear propuesta por Chen, Fu, y Rodrigues

En (Chen *et al.*, 2002) se propone una matriz, que representa una aproximación discreta de las formas de las piezas a ser ubicadas. La codificación consiste en que se cuadricula todo la placa disponible para ubicar las piezas, cada cuadro tiene dimensiones de 1x1 e inicialmente la matriz que representa la placa está llena de ceros lo que indica que no

hay espacios ocupados, a medida que se van ubicando las piezas los valores de los cuadros que representan la pieza. La figura 5.8 ilustra la matriz y como se realiza el almacenamiento de piezas en esta.

0	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	2	0	0	3	3
0	0	0	2	2	0	0	0
0	0	0	2	2	0	0	0

Figura 5.8. Matriz de ubicación de las piezas.

En la figura 5.9 se muestra representación rectilínea de la matriz, que generan un patrón de empaquetamiento.

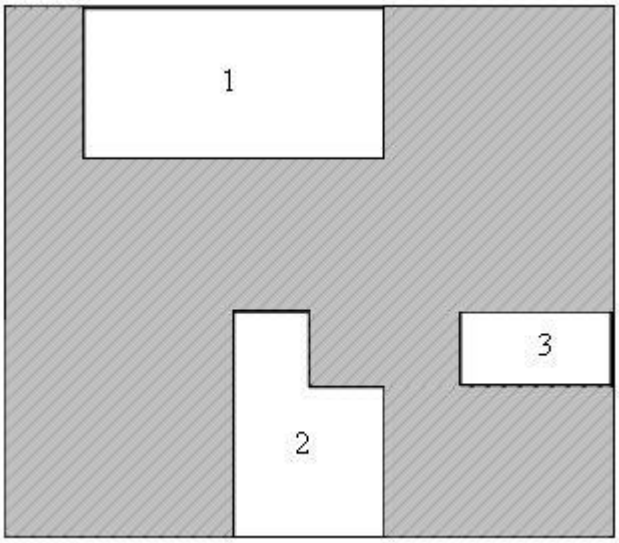

Figura 5.9. Representación rectilínea en la placa.

Algunas ventajas de usar esta representación son:

- La representación de cualquier forma geométrica, incluyendo piezas cóncavas y convexas (ver figura 5.9).
- La fácil verificación de rotación y traslapes de piezas.
- Pueden ser aplicadas heurísticas y metaheurísticas.

Algunas desventajas de esta representación son:

- La complejidad de la representación y el crecimiento computacional cuadrático con el tamaño de cada objeto.
- Únicamente son posibles cuatro orientaciones (arriba, abajo, derecha, izquierda)

- Para evaluar la función objetivo se contabilizan el número de espacios sin ocupar para así calcular el porcentaje de utilización del tablero y el valor del área sin ocupar.

En este estudio se seleccionó una representación presentada por Wong et al. en [76], esta estructura de datos codifica a través de un árbol (llamado árbol de cortes) los patrones de empaquetamiento del problema. Una de las grandes ventajas de la representación en árbol de cortes es que este genera solo patrones de corte tipo guillotina. Además, asegura que para todo el espacio de soluciones que conforman los diferentes patrones guillotina existe al menos un árbol de cortes que representa una de estas soluciones (Wong *et al.*, 1988). Diferentes metodologías propuestas han corroborado la efectividad de usar una codificación en árbol de cortes en especial la presentada por (Cui, 2007) y la de (Toro *et al.*, 2008).

La aplicación de técnicas de conglomerados se reduce a agrupar las piezas a cortar en grupos de piezas iguales, por lo que en los problemas donde no se permite la rotación de piezas este proceso es trivial, mientras que en los problemas donde la rotación de piezas de 90° es permitida se deben generar las diferentes $2n$ piezas (donde n es el número de piezas disponibles) y luego realizar la agrupación.

5.5. Cálculo de la función objetivo

En los subespacios generados por el árbol de cortes se debe realizar la ubicación de las piezas, este proceso se realiza mediante el algoritmo constructivo mejor-ajuste *(best-fit)*, esto hace que se conserven las restricciones tipo guillotina y sea embalada la mayor cantidad de piezas por subespacio (sin pesos) ó el conjunto de piezas que genere mejor beneficio (con pesos).

El algoritmo constructivo *best-fit* consiste en encontrar el conjunto de piezas idénticas que maximiza el área (maximiza el beneficio para versión con pesos) del subespacio j, sujeto a la restricción del número de piezas i disponibles. La ecuación (5.3) expresa formalmente el algoritmo *best-fit*.

$$\max \min_{i} \frac{W_j}{w_i} \bullet \frac{L_j}{l_i} \quad \text{, piezas disponibles tipo } i \quad {}_{i}{}_{i} \cdot w \cdot l \;; \forall i; i = 1, 2, \ldots, n \tag{5.3}$$

El cálculo de la función objetivo consiste en aplicar el algoritmo constructivo *best-fit* a cada subespacio, siendo el valor de la función objetivo la suma de las áreas de las piezas empacadas. El diagrama de flujo de datos del cálculo de la función objetivo es ilustrado en la figura 5.10.

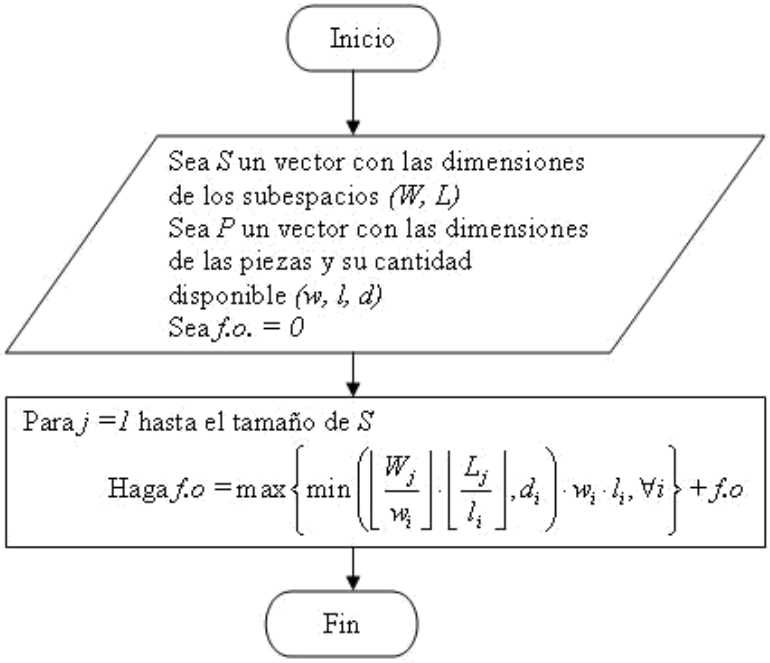

Figura 5.10. Algoritmo Cálculo de la función objetivo.

Por otro lado, para la versión con pesos es utilizada la ecuación 5.4.

$$\max \min \left\{ \left\lfloor \frac{W_j}{w_i} \right\rfloor \cdot \left\lfloor \frac{L_j}{l_i} \right\rfloor, \text{piezas disponibles tipo } i \right\} \cdot c_i \quad ;\forall i\,; i = 1, 2, \ldots, n \tag{5.4}$$

5.6. Metodología

La codificación propuesta en este capítulo garantiza la factibilidad en cuanto a las restricciones de corte tipo guillotina. Este tipo de codificación permite transformar los problemas de la mochila en problemas de minimización irrestrictos.

En este estudio se propone dividir el árbol de cortes en dos independientes entre sí: el árbol de orientación de cortes y el árbol de distancias de los cortes, representados por las variables *O* y *D* respectivamente, esto significa que para cada conjunto de valores de *O* existe una solución óptima D^*. El esquema de optimización para el problema es ilustrado en la figura 5.11, en este el algoritmo *I* realiza una búsqueda exhaustiva sobre el árbol *O,* mientras que el algoritmo *II* recibe todos árboles *O* posibles, donde debe encontrar las distancias óptimas para cada uno de estos. El algoritmo *II* corresponde al escenario de las técnicas metaheurísticas.

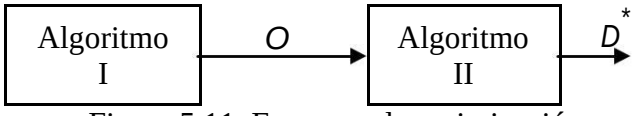

Figura 5.11. Esquema de optimización.

Algoritmo *I*

Como fue enunciado el algoritmo *I* genera los posibles árboles de orientación de los cortes, Los problemas de empaquetamiento en placas se restringe a solo árboles

completos de 3 niveles, el resultado de este algoritmo siempre dará como resultado 128 diferentes árboles de orientación, cada uno de estos es usado como dato de entrada para el algoritmo *II.*

Algoritmo *II*

En el algoritmo *II* se hace uso de las técnicas metaheurísticas de optimización, este combina las principales características de cúmulo de partículas, recocido simulado y algoritmos genéticos. El primero es el algoritmo principal, el segundo y tercero se usan como mecanismo de perturbación especializado para realizar búsquedas locales, efectuando cambios en las posiciones de las partículas usando la filosofía de la mutación de los genéticos y la temperatura del recocido simulado.

Algoritmo híbrido cúmulo de partículas, algoritmos genéticos y recocido simulado ($A_{PSO+GA+SA}$)

En este algoritmo se incluye el operador mutación propio de los algoritmos genéticos (Gallego *et al.*, 2008) en el algoritmo PSO, donde la mutación se define como la modificación del valor de un nodo del árbol de distancias a través del mecanismo de transición de la ecuación (5.5).

El mecanismo de transición consiste en permitir grandes cambios en las distancias de los cortes durante las primeras iteraciones, al igual que el recocido simulado permite empeoramientos de la función objetivo al comienzo del proceso. A medida que avanzan las iteraciones los cambios se vuelven más sensibles y determinísticos. La ecuación (5.5) está compuesta por: el valor actual del nodo *i* del árbol de distancias, el número de iteración actual, el número de iteraciones totales y un Epsilon (donde ε, es el mínimo porcentaje para generar un cambio en la distancias) donde, $\varepsilon = 100/\max(L,W)$.

$$nodo\ i = nodo\ i + \ rand - \frac{1}{2} \sqrt{1 - \frac{k}{IteracionesTotales}} + \varepsilon \qquad (5.5)$$

Dado que el algoritmo PSO presenta algunas similitudes con los algoritmos evolutivos como los algoritmos genéticos, diferentes autores han propuesto la inclusión del operador mutación en el algoritmo PSO.

Estas operaciones híbridas normalmente se implementan en cada generación (Andrews, 2006), (Carlisle y Dozier, 2000) o en un intervalo prefijado (Liang y Suganthan, 2005) o son controladas por una función de adaptación definida. En este estudio, en cada generación del algoritmo PSO, la población tiene la probabilidad de utilizar el operador de mutación.

La figura 5.12 ilustra el diagrama de flujo de datos del algoritmo $A_{PSO+GA+SA}$.

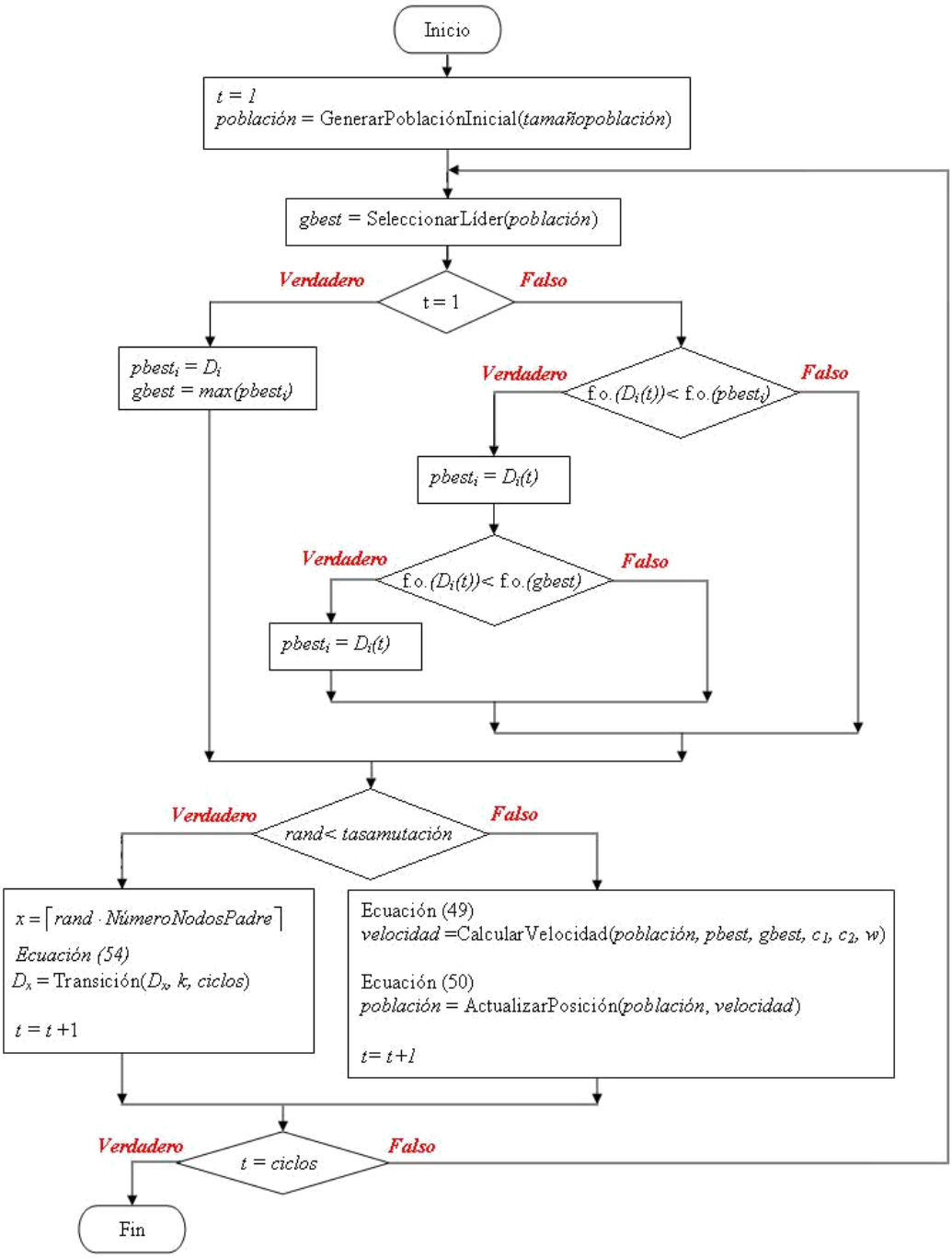

Figura 5.12. Diagrama del algoritmo $A_{PSO+GA+SA}$.

5.7. Calibración de parámetros

Una etapa de gran importancia en la aplicación de las técnicas metaheurísticas es el ajuste de parámetros. Diferentes enfoques se presentan para realizar la parametrización. (Pepper *et al.*, 2002) recaracterizan los parámetros del recocido simulado de forma tal que sin importar el tamaño del problema la selección de parámetros permita encontrar soluciones de buena calidad. Los parámetros del algoritmo implementado son presentados en la tabla 5.1.

Algoritmo	Parámetros
APSO+GA+SA	Tamaño de la Población Número de Ciclos c1 (Conocimiento Individual) c2(Conocimiento Grupal) w (Inercia) Tasa de Mutación Número de Niveles

Tabla 5.1. Parámetros del algoritmo implementado.

El problema crítico es que se pueden encontrar unos parámetros que presentan respuestas de excelente calidad para un tamaño en especial del problema y que al usar estos mismos en problemas más pequeños o más grandes se ve reducida la calidad de las respuestas y a su vez la eficiencia del algoritmo.

En general no existe un método exacto y eficiente para realizar la calibración de parámetros de las diferentes técnicas metaheurísticas, comúnmente estos algoritmos son parametrizados a través de la combinación de una búsqueda exhaustiva y una análisis estadístico de la calidad de los resultados. Para realizar la parametrización distintos estudios proponen: clasificar los problemas de prueba (si existen) por complejidad (matemática o computacional), escoger una problema representante (candidato) de cada clase, realizar un ajuste de los parámetros para cada candidato a través de una búsqueda exhaustiva en malla y por último recombinar los parámetros obtenidos para cada clase escogiendo la mejor combinación de estos.

(Zhi-Hui *et al*., 2009) presenta un rango de valores reducido para los parámetros del algoritmo PSO. Teniendo como base los rangos presentados por Zhi-Hui el tamaño de la malla se reduce considerablemente.

En este estudio se conserva la filosofía de los operadores de mutación de los algoritmos genéticos, donde la probabilidad de que ocurra una mutación en la población es muy baja. Para esto se realizó el mismo proceso de calibración para los rangos propuestos del parámetro de mutación en (Gallego *et al*., 2008).

Los valores resultantes de la calibración de parámetros son ilustrados en la tabla 5.2.

Algoritmo	Parámetros	Valor Problemas Con Pesos	Valor Problemas Sin Pesos
APSO+GA+SA	Tamaño de la Población	200	300
	Número de Ciclos	500	1000
	c1 (Conocimiento Individual)	2,05	2,05
	c2(Conocimiento Grupal)	2,05	2,05
	w (Inercia)	0,7	0,7
	Tasa de Mutación	0,03	0,03
	Número de Niveles	3	3

Tabla 5.2. Valores de los parámetros de cada algoritmo.

5.8. Resumen

En este capítulo se presentó las heurísticas, codificaciones, el algoritmo propuesto y la calibración de parámetros que conforman la metodología de solución propuesta en este trabajo. El uso de una codificación en árbol de cortes, el cálculo de la función objetivo utilizando la heurística *best-fit*, la selección de tres técnicas de optimización para generar un algoritmos híbrido de optimización y un proceso cuidadoso de calibración de parámetros de los algoritmos representan la metodología de solución de los diferentes problemas de empaquetamiento propuestos en este estudio.

Utilizar estructuras de datos tipo árbol de cortes garantiza las restricciones de corte tipo guillotina y presenta un escenario propicio para las técnicas metaheurísticas de optimización propuestas. Dado que los árboles de cortes no representan la ubicación de las piezas sobre el material, una rutina debe realizar este proceso además de medir la calidad de la ubicación propuesta. En este estudio esta rutina es llamada cálculo de la función objetivo. Realizar el cálculo de la función objetivo utilizando la heurística *best-fit*, permite agrupar las piezas mediante el criterio de igualdad de sus formas, lo que potencia el uso del árbol de cortes.

En este estudio se propone un algoritmo híbrido $A_{PSO+GA+SA}$ para optimizar el árbol de distancias de los cortes, inspirados en tres técnicas metaheurísticas (PSO, GA y SA).

Calibrar los parámetros de las técnicas metaheurísticas es un proceso importante dado que este afecta directamente la calidad y el tiempo de respuesta de los algoritmos. En este estudio se realiza un ajuste detallado y cuidadoso de todos los parámetros, mediante una búsqueda exhaustiva en malla en rangos de valores sugeridos en la literatura especializada.

Un análisis estadístico permitirá verificar el desempeño en la solución de los diferentes problemas propuestos en este estudio. En el siguiente capítulo se describe el estudio computacional realizado del algoritmo de solución sobre los problemas propuestos.

Capítulo 6.

ANÁLISIS DE RESULTADOS

6.1. Introducción

En este estudio se realiza un análisis estadístico entorno a la calidad de la solución y tiempos de respuesta del algoritmo propuesto, utilizando como descriptores estadísticos la media y la desviación estándar para representar el comportamiento de la calidad de respuesta del algoritmo implementado para cada problema de este estudio.

Se describen los casos de prueba utilizados, los detalles de la implementación de los algoritmos, los resultados computacionales y por último se realiza un análisis comparativo de los resultados obtenidos.

6.2. Casos de prueba

Dado que en este estudio se tiene en cuenta todos los posibles surtidos de piezas, las librerías de casos de prueba seleccionadas deben cumplir con todas las diferentes características necesarias, las librerías utilizadas para los diferentes problemas son variados en cuanto a la complejidad matemática y son diseñados especialmente para cada tipo de problema.

Fueron seleccionados 30 casos de prueba para el problema de la mochila bidimensional guillotinada, 15 casos para la versión con pesos ([UW1-UW11] y [UWL1-UWL4]) y 15 casos para la versión sin pesos ([UU1-UU11] y [UUL1-UUL4]). Estos casos presentan 30 tipos de mochilas diferentes con distribuciones entre 25 y 200 piezas, la base de datos es presentada por (Hifi, 2001) y disponible en línea en (Hifi, 1997). De la totalidad de casos 22 ([UW1-UW11] y [UU1-UU11]) pertenecen a la categoría de problemas de empaquetamiento de mediana complejidad matemática y 8 ([UWL1-UWL4] y [UUL1-UUL4]) de alta complejidad matemática. Diferentes estudios han utilizado estos casos de prueba para realizar una especie de benchmark de las metodologías propuestas.

Existe gran cantidad de librerías para los problemas presentados en este estudio, pero luego de ser analizadas muchas caen en la categoría de problemas de empaquetamiento perfecto, es decir, los casos prueba son desarrollados a partir de la solución óptima del problema, lo cual hace que metodologías exactas parezcan un buen método de solución.

Las bases de datos utilizadas en este estudio no necesariamente presentan un óptimo global en su límite inferior, que hacen que las metodologías exactas renuncien en su proceso de optimización debido al costo computacional.

6.3. Detalles de la implementación

Todos los algoritmos fueron desarrollados en Delphi 7,0 ®, sobre un ordenador con unas especificaciones de un procesador Pentium 4 de 3,0 GHz y una memoria RAM de 1 GB.

6.4. Resultados computacionales

Los sistemas de prueba usados en este estudio fueron tomados de la literatura especializada después de realizar una revisión bibliográfica. Las metodologías usadas en la solución de dichos problemas son resueltos con métodos aproximados o exactos. Los problemas seleccionados son variados en cuanto a la complejidad matemática y tipo de problema a solucionar.

Se presentan para todos los casos de prueba de cada tipo de problema la mejor solución reportada en la literatura especializada (*Best Known Solution*). Los resultados de (Hifi, 1998) presentan las mejores soluciones aunque trabajos como (G *et al*., 2003) alcanzan buenos resultados. Casos como UWL1, UWL2, UWL3, UWL4, UUL1, UUL2, UUL3 y UUL4 para el problema de la mochila sin rotación no tiene respuesta reportada debido a que la mayoría de aproximaciones a estos casos de gran complejidad matemática fueron realizadas mediante el uso de técnicas exactas donde el esfuerzo computacional es demasiado alto para su solución. Por lo tanto, algunos autores omiten reportarlas.

Las tablas 6.1 y 6.2 presentan los mejores resultados reportados en la literatura especializada y su respectivo autor.

Caso	*Best Known Solution*	Tiempo Utilizado (Segundos)	Caso	*Best Known Solution* / *H ×W*	Tiempo Utilizado (Segundos)	Autor
UW1	6036	13	UU1	97,17	372	(Hifi, 2001)
UW11	15747	84	UU2	99,21	13	(Hifi, 2001)
UW2	8468	53	UU3	97,52	19	(Hifi, 2001)
UW3	6226	32	UU4	98,25	67	(Hifi, 2001)
UW4	8326	135	UU5	99,15	106	(Hifi, 2001)
UW5	7780	58	UU6	98,79	43	(Hifi, 2001)
UW6	6615	177	UU7	98,84	255	(Hifi, 2001)
UW7	10464	301	UU8	98,98	168	(Hifi, 2001)
UW8	7692	452	UU9	99,2	160	(Hifi, 2001)
UW9	7038	258	UU10	99,01	>10800	(Hifi, 2001)
UW10	7507	932	UU11	99,85	>10800	(Hifi, 2001)
UWL1		>10800	UUL1		>10800	(Hifi, 2001)
UWL2		>10800	UUL2		>10800	(Hifi, 2001)
UWL3		>10800	UUL3		>10800	(Hifi, 2001)
UWL4		>10800	UUL4		>10800	(Hifi, 2001)

Tabla 6.1. Mejor solución reportada con y sin pesos, sin rotación.

Para realizar un estudio estadístico de la calidad de las respuestas obtenidas, los casos de prueba fueron ejecutados 20 veces. Como índice de calidad de las respuestas se utilizó el error porcentual, este se define como el porcentaje de desviación de la respuesta obtenida por el algoritmo con respecto a la mejor solución reportada en la literatura. La ecuación (6.1) define formalmente el error porcentual. Los descriptores estadísticos de la calidad de la solución de cada algoritmo son la media del error (error medio) y la desviación estándar del error. El error medio se calcula como el error promedio de la muestra y la desviación estándar del error se define como la dispersión de los valores respecto al error medio.

Caso	*Best Known Solution*	Tiempo Utilizado (Segundos)	Caso	*Best Known Solution* $H \times W$	Tiempo Utilizado (Segundos)	Autor
UW1	6696	29	UU1	98,42	17	(Hifi, 2001)
UW11	18200	138	UU2	99,28	49	(Hifi, 2001)
UW2	9732	124	UU3	99,03	3437	(Hifi, 2001)
UW3	7188	111	UU4	99,05	186	(Hifi, 2001)
UW4	8452	4870	UU5	99,64	352	(Hifi, 2001)
UW5	8398	196	UU6	98,81	263	(Hifi, 2001)
UW6	6937	635	UU7	99,46	>10800	(Hifi, 2001)
UW7	11585	1104	UU8	99,37	444	(Hifi, 2001)
UW8	8088	921	UU9	99,36	>10800	(Hifi, 2001)
UW9	7527	1570	UU10	99,35	2239	(Hifi, 2001)
UW10	8172	1283	UU11	99,88	>10800	(Hifi, 2001)
UWL1	171079321	913	UUL1	99,99	704	(Hifi, 2001)
UWL2	325725070	1281	UUL2	99,99	1336	(Hifi, 2001)
UWL3	433859655	3095	UUL3	99,99	4165	(Hifi, 2001)
UWL4	568436545	6017	UUL4	99,99	6170	(Hifi, 2001)

Tabla 6.2. Mejor solución reportada con y sin pesos, con rotación.

$$error = \frac{BestKnownSolution - SoluciónObtenida}{BestKnownSolution} \qquad (6.1)$$

Los problemas propuestos tendrán un valor de error medio y desviación estándar, que representará la calidad de sus respuestas. Además de esto, interesa el mejor valor de cada muestra (mejor solución alcanzada por muestra). La tabla 6.3 resume los errores medios y las desviaciones estándar para cada problema. Mientras las tablas 6.4 y 6.5 presentan el mejor valor en cada muestra.

Problema	Error Medio	Desviación Estándar
Con pesos Sin rotación	0,03315	0,01244
Sin pesos Sin rotación	0,04975	0,02171
Con pesos Con rotación	0,03655	0,01711
Sin pesos Sin rotación	0,04740	0,01841

Tabla 6.3. Error medio y desviación estándar para cada problema.

En las tablas 6.4 y 6.5 se utilizan diferentes símbolos para representar la calidad de las respuestas respecto a las reportadas en la literatura, se utiliza la siguiente codificación para leer los resultados:

α representa una respuesta igual a la reportada en la literatura especializada.
β representa una respuesta mejor que la reportada en la literatura especializada.
γ representa una respuesta que no supera a las reportadas en la literatura especializada.

Caso	Solución Propuesta $A_{PSO+GA+SA}$	Tiempo Utilizado (Segundos)	Caso	$\frac{Solución\ A_{PSO+GA+SA}}{H \times W}$	Tiempo Utilizado (Segundos)
UW1	6036 $^{\alpha}$	33	UU1	97,17 $^{\alpha}$	106
UW11	15747 $^{\alpha}$	37	UU2	99,21 $^{\alpha}$	120
UW2	8468 $^{\alpha}$	45	UU3	96,82 $^{\gamma}$	106
UW3	6226 $^{\alpha}$	56	UU4	97,39 $^{\gamma}$	145
UW4	8326 $^{\alpha}$	59	UU5	99,15 $^{\alpha}$	186
UW5	7780 $^{\alpha}$	45	UU6	98,79 $^{\alpha}$	145
UW6	6615 $^{\alpha}$	59	UU7	98,61 $^{\gamma}$	186
UW7	10464 $^{\alpha}$	62	UU8	98,98 $^{\alpha}$	212
UW8	7692 $^{\alpha}$	69	UU9	99,2 $^{\alpha}$	231
UW9	7038 $^{\alpha}$	56	UU10	98,3 $^{\gamma}$	212
UW10	7507 $^{\alpha}$	74	UU11	99,73 $^{\gamma}$	105
UWL1	158363374 $^{\beta}$	155	UUL1	99,95 $^{\beta}$	280
UWL2	257372669 $^{\beta}$	158	UUL2	99,96 $^{\beta}$	282
UWL3	261252077 $^{\beta}$	320	UUL3	99,98 $^{\beta}$	320
UWL4	473754165 $^{\beta}$	450	UUL4	99,98 $^{\beta}$	420

Tabla 6.4. Mejores resultados $A_{PSO+GA+SA}$ para la mochila con y sin pesos, sin rotación.

Caso	Solución Propuesta $A_{PSO+GA+SA}$	Tiempo Utilizado (Segundos)	Caso	$\frac{Solución\ A_{PSO+GA+SA}}{H \times W}$	Tiempo Utilizado (Segundos)
UW1	6696 $^{\alpha}$	65	UU1	98,42 $^{\alpha}$	210
UW11	18200 $^{\alpha}$	72	UU2	99,28 $^{\alpha}$	238
UW2	9732 $^{\alpha}$	88	UU3	99,03 $^{\alpha}$	210
UW3	7188 $^{\alpha}$	100	UU4	98,97 $^{\gamma}$	288
UW4	8452 $^{\alpha}$	116	UU5	99,64 $^{\alpha}$	374
UW5	8398 $^{\alpha}$	89	UU6	98,81 $^{\alpha}$	288
UW6	6937 $^{\alpha}$	116	UU7	98,67 $^{\gamma}$	374
UW7	11585 $^{\alpha}$	122	UU8	98,98 $^{\gamma}$	422
UW8	8088 $^{\alpha}$	136	UU9	99,2 $^{\gamma}$	460
UW9	7527 $^{\alpha}$	110	UU10	99,16 $^{\gamma}$	422
UW10	8172 $^{\alpha}$	147	UU11	99,86 $^{\gamma}$	210
UWL1	171836653 $^{\beta}$	302	UUL1	99,99 $^{\alpha}$	500
UWL2	326399541 $^{\beta}$	299	UUL2	99,99 $^{\alpha}$	503
UWL3	433882185 $^{\beta}$	610	UUL3	99,99 $^{\alpha}$	604
UWL4	568690356 $^{\beta}$	700	UUL4	99,99 $^{\alpha}$	750

Tabla 6.5. Mejores resultados $A_{PSO+GA+SA}$ para la mochila con y sin pesos, con rotación.

La tabla 6.6 presenta un resumen de las tablas 6.4 y 6.5. Esta tabla compara el mejor valor encontrado por los algoritmos para cada instancia del problema con la mejor solución reportada. De esta comparación se puede dar que la mejor respuesta supere la mejor respuesta reportada, que no la supere o que la iguale.

	Problema de la Mochila							
	Con Pesos				Sin Pesos			
	Sin Rotación		Con Rotación		Sin Rotación		Con Rotación	
	Calidad	Tiempo	Calidad	Tiempo	Calidad	Tiempo	Calidad	Tiempo
Igualados	11		11		6		9	
Superados	**4**	**13**	**4**	**14**	**4**	**8**		**11**
Inferiores		2		1	5	7	6	4

Tabla 6.6. Comparación de resultados de los mejores resultados.

La figura 6.1 representa gráficamente los valores de los descriptores estadísticos (media y desviación estándar) de la calidad de solución en los diferentes problemas.

La tabla 6.7 presenta los tiempos computacionales requeridos por el algoritmo para cada problema. En esta se ilustran dos tipos de tiempos: totales y promedios por conjunto de problemas.

El tiempo total se definen como el tiempo necesario para el algoritmo resolver todos los casos de prueba 20 veces, mientras el tiempo promedio se define como el tiempo necesario para resolver todos los casos de prueba con igual complejidad (mediana o gran escala) sobre el número total de casos de prueba.

La figura 6.2 ilustra los tiempos totales utilizados por el algoritmo para resolver cada problema.

Problema	**Tiempos**	**APSO+GA+SA**
Con Pesos Sin Rotación	Tiempo total	33.560
	Tiempo promedio (mediana escala)	54
	Tiempo promedio (gran escala)	271
Con Pesos Con Rotación	Tiempo total	61.440
	Tiempo promedio (mediana escala)	106
	Tiempo promedio (gran escala)	478
Sin Pesos Sin Rotación	Tiempo total	61.120
	Tiempo promedio (mediana escala)	159
	Tiempo promedio (gran escala)	326
Sin Pesos Con Rotación	Tiempo total	117.060
	Tiempo promedio (mediana escala)	318
	Tiempo promedio (gran escala)	589

Tabla 6.7. Tiempos utilizados por cada algoritmo (segundos).

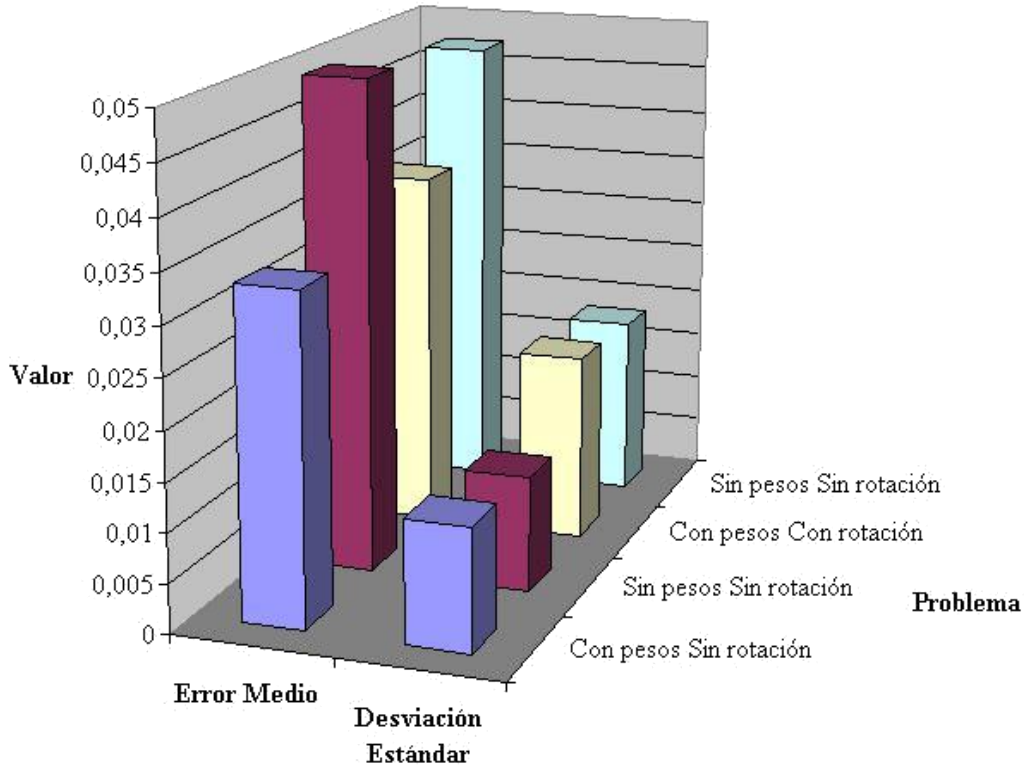

Figura 6.1. Gráfica error medio y desviación estándar.

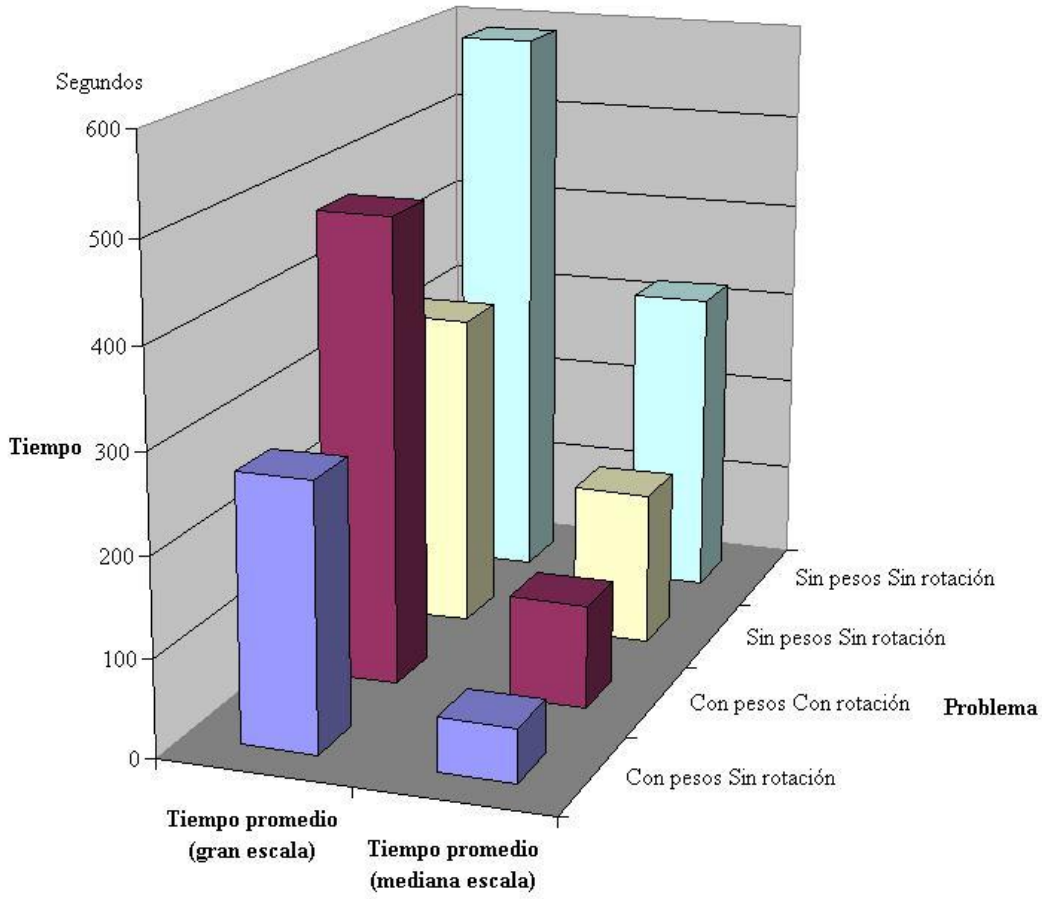

Figura 6.2. Gráfica de tiempos promedios (segundos).

6.5. Análisis

En las tablas 6.4 y 6.5 se resumen los mejores resultados alcanzados por la metodología propuesta. En la tabla 6.6 se resumen los resultados comparados con los reportados en la literatura especializada. De la tabla 6.6 se logra identificar que la metodología propuesta para resolver los problemas de mochila con pesos asociados, con o sin rotación supera en un 27% las mejores respuestas reportadas en la literatura especializada y en el 73% restante las iguala.

La metodología propuesta aplicada a los problemas de la mochila sin pesos asociados presenta un comportamiento regular, debido a que en los problemas donde no se permite la rotación de piezas solo el 27 % de las respuestas reportadas son superadas, el 40% de las respuestas son igualadas y en el 33% restante no se logra alcanzar las reportadas. En los problemas donde se permite la rotación de piezas un 60% de las soluciones igualan las reportadas y en el 40% restante no se logran alcanzar las reportadas.

Al analizar la tabla 6.7, los tiempos de respuestas de la metodología propuesta son excelentes para resolver los problemas de la mochila con pesos de gran escala (con distribuciones de hasta 200 piezas) sin rotación en un tiempo promedio de 271 segundos, problemas de la mochila con pesos de mediana escala con y sin rotación de piezas (hasta 60 piezas) tiempos promedios de 106 y 54 segundos respectivamente y problemas de la mochila sin pesos de mediana escala con y sin rotación en tiempos promedios de 158 y 318 segundos respectivamente. Sin embargo, en los problemas de de la mochila con y sin pesos de gran escala (hasta 100 piezas) con rotación presentan unos tiempos promedios relativamente altos de 8 y 10 minutos respectivamente.

En la tabla 6.3 se identifica que la metodología propuesta para los problemas de la mochila tiene un error medio 4,1% respecto a la mejor respuesta reportada, lo que significa que para cualquier problema de mochila con o sin pesos asociados, con o sin rotación la solución obtenida con la metodología propuesta está muy cerca del óptimo global, dado que en aplicaciones reales se manejan en el proceso de corte o empaquetamiento unos límites de desperdicio del 30%. En este estudio se obtiene una reducción en las pérdidas del material de más del 50%. El inconveniente en la aplicación de estas metodologías es el alto tiempo de cómputo.

Los tiempos de cómputo utilizados por la metodología son menores a los reportados en la literatura especializada pero debido a las diferencias entre las arquitecturas de cómputo y lenguajes de programación no se puede concluir entre metodologías. En particular los tiempos de cómputo utilizados por la metodología propuesta son razonables.

Además, con la metodología desarrollada se alcanzan resultados de mejor calidad que los reportados a través de métodos exactos en la literatura especializada.

6.6. Resumen

Se presentaron los casos de prueba utilizados en este estudio, las especificaciones del hardware y software sobre los cuales se ejecutaron los algoritmos propuestos, los

resultados obtenidos para realizar el análisis estadístico, tiempos de respuesta de la metodología y análisis de resultados.

Se efectuó un análisis estadístico con el fin de medir la calidad del algoritmo implementado sobre cada problema, utilizando la media y la desviación estándar del error (diferencia entre la solución alcanzada y la mejor solución reportada en la literatura) como descriptores estadísticos y definiendo un tamaño de muestra de 20 ejecuciones por instancia de prueba. Además fueron medidos los tiempos de ejecución utilizados por el algoritmo.

La metodología propuesta presenta un error de un 4,1% para cualquier problema de la mochila y logra superar algunas respuestas reportadas en la literatura.

Capítulo 7.

CONCLUSIONES Y RECOMENDACIONES

En esta investigación se estudió un algoritmo híbrido de optimización metaheurística para solucionar los problemas de: la mochila bidimensional irrestricta con patrones de corte tipo guillotina (*unconstrained two-dimensional guillotineable knapsack problem*), con y sin pesos asociados a las piezas, con y sin rotación de 90° de las piezas.

Se realizó la revisión bibliográfica de los modelos matemáticos de los problemas de la mochila bidimensional irrestricta con patrones de corte tipo guillotina, con y sin pesos asociados a las piezas, con y sin rotación de piezas. Los problemas propuestos en esta tesis has sido estudiados por más de 6 décadas sin embargo no se ha llegado a un consenso general que determine un modelo matemático definido y que incluyan las diferentes características que den solución al problema de la vida real, en especial los patrones de corte guillotina. En este trabajo se adaptó un modelo propuesto recientemente por (Ben *et al.*, 2008), para describir los problemas presentados en este estudio, aunque carece de la restricción de rotación de las piezas 90°.

Se realizó una revisión bibliográfica de las diferentes técnicas metaheurísticas de optimización propuestas para resolver los problemas de de la mochila bidimensional irrestricta. Con base en las diferentes técnicas de optimización estudiadas en este trabajo se propuso un algoritmo híbrido, resultante de la combinación entre estas. En esta tesis se utilizaron tres técnicas metaheurísticas (cúmulo de partículas, algoritmo genético y recocido simulado), con base en estos se generó un algoritmo híbrido de optimización que reúne las principales características de cada uno.

Se realizó una revisión de las diferentes codificaciones propuestas para los problemas de de la mochila bidimensional irrestricta. En esta tesis se optó por una codificación en árbol de cortes, dividiendo este en dos árboles uno de orientación de los cortes y otro de distancias de cortes.

Se desarrolló una metodología de solución que consiste en realizar una búsqueda exhaustiva sobre el árbol de orientación de los cortes y un algoritmo híbrido que combina tres técnicas metaheurísticas de optimización para encontrar las distancias de cortes óptimas del árbol.

El algoritmo propuesto presenta resultados de excelente calidad al resolver casos de prueba presentados en la literatura especializada. Aunque este incluye una gran cantidad de parámetros que deben ser ajustados y emplea un alto tiempo de cómputo.

El algoritmo propuesto explota las principales propiedades de cada técnica metaheurística de acuerdo a la codificación de árbol de cortes para mejorar los procesos de exploración y explotación que son la base de los procesos búsqueda de las técnicas combinatorias de optimización.

Trabajos futuros propuestos en esta investigación.

Utilizar algoritmos paralelizables en la solución de los problemas estudiados, con el fin de mejorar tiempos y calidad de las respuestas.

Implementar el algoritmo propuesto en la solución de los problemas de empaquetamiento óptimo bidimensional guillotinado en una sola placa, en placas y en rollos infinitos con y sin rotación de piezas.

Adaptar la codificación propuesta en el estudio problemas de empaquetamiento óptimo tridimensional guillotinado y extruir la codificación para el problema de empaquetamiento óptimo de cajas en contenedores usando paletas.

Usar otras técnicas metaheurísticas de optimización utilizando la codificación propuesta con el fin de mejorar la calidad de las respuestas.

Implementar una metodología exacta para realizar el ajuste óptimo de parámetros de las técnicas metaheurísticas de optimización propuestas en este trabajo.

Utilizar esta metodología de solución en problemas de la vida real, realizando las adaptaciones necesarias y haciendo uso de información real.

REFERENCIAS

ANDREWS, P. S. An investigation into mutation operators for particle swarm optimization. *in Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, pp. 1044–1051, 2006.

BEASLEY J. E. "Algorithms for unconstrained two-dimensional guillotine cutting". *Journal of the Operational Research Society*, vol. 36, pp. 297-306, 1985.

BEASLEY, J. E. "An exact two-dimensional non-guillotine cutting tree search procedure*". Oper. Res.*, vol. 33, pp. 49–64, 1986.

BEASLEY, J. E. "A population heuristic for constrained two-dimensional nonguillotine cutting". *European Journal of Operational Research*, vol. 156, pp. 601- 627, 2004.

BEN, S. CHU, C. y ESPINOUSE, M. L. "Characterization and modelling of guillotine constraints". *European Journal of Operational Research*, vol. 191, pp. 112–126, 2008.

BIRO, M. y BOROS, E. "Network flows and non-guillotine cutting patterns". *European Journal of Operational Research*, vol. 16, pp. 215–221, 1984.

BISCHOFF, E. E. y MARRIOTT, M.D., "A comparative evaluation of heuristics for container loading". *European Journal of Operational Research*, vol. 44, pp. 267–276, 1990.

BORTFELDT, A. y GEHRING, H. "A hybrid genetic algorithm for the container loading problem". *European Journal of Operational Research*, vol. 131, pp. 143–161, 2001.

CAPRARA, A. y MONACI, M. "On the two-dimensional knapsack problem". *Operations Research Letters*, vol. 32, pp. 5–14, 2004.

CARLISLE, A. y DOZIER, G. Adapting particle swarm optimization to dynamic environments. *in Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, pp. 429–434, 2000.

CHEN, P.; FU, Z.; LIM, A. y RODRIGUES B. "Two-Dimesional Packing for irregular shaped objects". *36th Hawaii International Conference on System Sciencies*. Computer Society. IEEE. 2002.

CHEN, C. S. LEE, S. M. y SHEN, Q. S. "An analytical model for the container loading problem". *European Journal of Operational Research*, vol. 80, pp. 68–76, 1995.

CHAZELLE, B. “The bottom-left bin packing heuristic: An efficient implementation”. *IEEE Transactions on Computers*, vol. 32, pp. 697–707, 1983.

CHRISTOFIDES, N. y WHITLOCK, C. “An algorithm for two-dimensional cutting problems”. *Operations Research*, vol. 25, pp. 31-44, 1977.

CUI, Y. An exact algorithm for generating homogenous T-shape cutting patterns. *Computers & Operations Research*, vol. 34, pp. 1107-1120, 2007.

DANIELS, K. MILENKOVIC, V. J. y LI, Z. “Multiple containement methods”. Technical Report 12-94, *Center for Research in Computing Technology*, Division of Applied Sciences, Harvard University, 1994.

DARWIN, C. “The Origin of Species”, *John Murray*, 1859.

FAYARD, D. y ZISSIMOPOULOS, V. “An approximation algorithm for solving unconstrained two-dimensional knapsack problems”. *European Journal of Operational Research*, vol. 84, pp. 618-632, 1995.

FEKETE, S. P. y SCHEPERS, J. “A new exact algorithm for general orthogonal d-dimensional knapsack problems”. In: Burkard, R., Woeginger, G.J. (Eds.), Algorithms-Proceedings of the 5th Annual European Symposium, Graz, Austria, Sept., 1997, *Lecture Notes in Computer Science*, vol. 1284. Springer, Berlin, pp. 144–156, 1997.

FEKETE, S. P. y SCHEPERS, J. “On more-dimensional packing III: Exact algorithms”. Technical Report ZPR97-290, *Mathematisches Institut*, UniversitYat zu KYoln, 1998.

G, Y.-G. y KANG, M.-K. “A new upper bound for unconstrained two-dimensional cutting and packing”. *J. Oper. Res. Soc.*, vol. 53, pp. 587–591, 2002.

G, Y.-G. KANG, M.-K. y SEONG, J. “A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems”. *Operations Research Letters*, vol. 31, pp. 301–307, 2003.

GALLEGO, R.; ESCOBAR, A. H. y ROMERO, R. A. *Programación lineal entera*, Textos universitarios, Universidad Tecnológica de Pereira, 2007.

GALLEGO, R.; ESCOBAR, A. H. y TORO, E. M. *Técnicas metaheurísticas de optimización*, Textos universitarios, Universidad Tecnológica de Pereira, 2008.

GAREY, M. R. y JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif, USA, 1979.

GILMORE, P. C. y GOMORY, R. E. “A linear programming approach to the cutting-stock problem”. *Operations Research*, vol. 9, pp. 849–859, 1961.

GILMORE, P. C. y GOMORY, R. E. “Multistage cutting stock problems of two and more dimensions”. *Operations Research*, vol. 13, pp. 94–120, 1965.

GILMORE, P. C. y GOMORY, R. E., "The theory and computation of knapsack functions". *Operations Research*, vol. 14, pp. 1045-1074, 1966.

HADJICONSYANTINOU, E. y CHRISTOFIDES, N. "An exact algorithm for general, orthogonal, two dimensional knapsack problems". *European Journal of Operacional Research*, vol. 83, pp. 39-56, 1995.

HEALY, P. y MOLL, R. "A local optimization-based solution to the rectangle layout problem". *Journal of the Operational Research Society*, vol. 47, pp. 523–537, 1996.

HERZ, J. C. "A recursive computing procedure for two-dimensional stock cutting". *IBM Journal of Research and Development*, vol. 16, pp. 462-469, 1972.

HIFI, M. "Exact algorithms for large-scale unconstrained two and three staged cutting problems". *Computational Optimization and Applications*, vol. 18, pp. 63–88, 2001.

HIFI, M. *Problem instances for the 2D Cutting/Packing Problems*, [web en línea], 1997. <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/> [Consulta: 01-02-2010].

HIFI, M. y ZISSIMOPOULOS, V. "A recursive exact algorithm for weighted two-dimensional cutting". *European J. Oper. Res.*, vol. 91, pp. 553–564, 1996.

JAKOBS, S. "Theory and methodology on genetic algorithms for the packing of polygons". *European Journal of Operational Research*, vol. 88, pp. 87-100, 1996.

KANTOROVICH, L. V. "Mathematical methods of organizing and planning production". *Management Science*, vol. 6, pp. 363-422, 1960.

KENNEDY, J. y EBERHART, R. C. "Particle Swarm Optimization". *In Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Piscataway, New Jersey, IEEE Service Center, 1995.

KIRKPATRICK, S. GELLATT, C. y VECCHI, M. "Optimization by Simulated Annealing". *Science*, vol. 220, pp. 671–680, 1983.

KRÖGER B., "Guillotineable bin packing: A genetic approach", *European Journal of Operational Research*, Vol. 84, pp. 645-661, 1995.

LIANG, J. J. y SUGANTHAN, P. N. Dynamic multi-swarm particle swarm optimizer with local search. *in Proc. IEEE Congr. Evol. Comput.*, pp. 522–528, 2005.

LODI, A. MARTELLO, S. y MONACI, M. "Two-dimensional packing problems: A Survey". *European Journal of Operational Research*, vol. 141, pp. 241–252, 2002.

LODI, A. MARTELLO, S. y VIGO, D. “Models and bounds for the two-dimensional level packing problems”. *Journal of Combinatorial Optimization*, vol. 8, pp. 363–379, 2004.

MARTELLO, S. MONACI, M. y VIGO, D. “An exact approach to the strip-packing problem,” *INFORMS Journal on Computing*, vol. 15, pp. 310–319, 2003.

MARTELLO, S. PISINGER, D. y TOTH, P. “New trends in exact algorithms for the 0-1 knapsack problem”, *European Journal of Operational Research*, vol. 123, pp. 325–332, 2000.

MARTELLO, S. y TOTH, P. *Knapsack Problems - Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A. y TELLER, E. “Equation of state calculations by fast computing machines”. *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

MORABITO, R. y ARENALES, M. “An graph approach to the solution of two dimensional non-guillotine cutting problems”. *European Journal of Operational Research*, vol. 84, pp. 599-617, 1995.

MORABITO, R. ARENALES, M. y ARCARO, V. “An and-or-graph approach for two-dimensional cutting problems”. *European Journal of Operational Research*, vol. 58, pp. 263-271, 1992.

MORABITO, R. y MORALES, A. “A simple and effective recursive procedure for manufacturer’s ballet loading problem”. *Journal of the Operational Research Society*, vol. 49, pp. 819-828, 1998.

ONODERA, H. TANIGUCHI, Y. y TAMARU, K. “Branch-and-bound placement for building block layout”. *in Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 433–439, 1991.

PEPPER, J.; GOLDEN, B. y WASIL, E. Solving the Traveling Salesman Problem with Annealing-Based Heuristics: A Computational Study. *IEEE Trans. Syst., Man, Cybern. A*, vol. 32, pp. 72-77, 2002.

PISINGER, D. “Heuristics for the container loading problem”. *European Journal of Operational Research*, vol. 141, pp. 382–392, 2002.

REYNOLDS, R. G. “An Introduction to Cultural Algorithms”. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pp. 131–139. World Scientific, River Edge, New Jersey, USA, 1994.

SCHEITHAUER, G. “LP-based bounds for the container and multi-container loading problem”. *International Transactions in Operational Research*, vol. 6, pp. 199– 213, 1999.

SCHEITHAUER, G. y TERNO, J. “Modeling of packing problems”. *Optimization*, vol. 28, pp.63-84, 1993.

TORO, E.; GARCÉS, A. y RUIZ, H. “Solución al problema de empaquetamiento bidimensional usando un algoritmo híbrido constructivo de búsqueda en vecindad variable y recocido simulado”. *Revista Facultad de Ingeniería Universidad de Antioquia*, vol. 46, pp. 119-131, 2008.

SWEENEY, P. y PATERNOSTER, R. “Cutting and Packing Problems: A categorized, application-orientated research bibliography”. *Journal of the Operational Research Society*, vol. 43, pp. 691-706, 1992.

WÄSCHER, G. HAUßNER H. y SCHUMANN H. “An improved typology of cutting and packing problems”. *European Journal of Operational Research*, vol. 183, pp. 1109–1130, 2007.

WONG, D. F.; LEONG, H. W. y LIU, C. L. *Simulated Annealing for VLSI Design*. Kluwer Academic Publishers, 1988.

ZHI-HUI, Z.; JUN, Z.; YUN, L. and Henry, S. Adaptive Particle Swarm Optimization. *IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics*, vol. 39, Issue 6, pp. 1362-1381, 2009.